

Metamodel-Driven Development of an Assurance Case Notation

Nungki Selviandro

PhD

University of York
Computer Science

January 2021

*For my parents,
my beloved wife Gia
and
my lovely sons Arka and Declan*

Abstract

The Structured Assurance Case Metamodel (SACM) is a standard specified by the Object Management Group that defines a metamodel for representing structured assurance cases. It is developed to support standardisation and interoperability in assurance case development. Unlike existing frameworks such as GSN and CAE, SACM (2.0) was not equipped with any visual notation that can be used to develop graphical assurance cases. A visual notation can be seen as an alternative representation to the textual form of a modelling language that can support the language's adoption by its users. The SACM concept is specified using a metamodel. We identified that there is a lack of a systematic approach that can be adopted to create a visual notation that considers the metamodel as part of the design process. This thesis proposes an approach that considers a metamodel structure as one of the inputs to developing a visual notation. The proposed approach could be used to develop any notation that is based upon a metamodel.

In this thesis, the application of the proposed approach is demonstrated in the development of the SACM notation. The produced SACM notation is evaluated through empirical studies involving novice and experienced users. The evaluation results shows that experienced users found the resulting SACM notation is more intuitive when compared to an existing notation. It is important to notice that the proposed approach requires many distinct design decisions when applying the approach to produce a notation. Thus, the development of the notation is sensitive to the design decision made by the notation designer that could affect the effectiveness of the resulting notation. Based on the evaluation results, the proposed approach is demonstrated to be helpful in developing a visual notation based on a defined metamodel. Furthermore, the resulting SACM notation is also shown to be useful, and able to be adopted by the practitioners to develop assurance cases. This is indicated by the adoption of the produced notation by an international standard body (Object Management Group); the SACM notation has been published as part of version 2.1 of the SACM standard.

Contents

Abstract	3
Acknowledgements	16
Declaration	17
1 Introduction	18
1.1 Motivation	18
1.2 Research questions	20
1.3 Thesis objectives and hypothesis	21
1.4 Thesis structure	21
2 Literature Survey	23
2.1 Introduction	23
2.2 Structured assurance case	23
2.2.1 Structure assurance case definition	24
2.2.2 Assurance case representations	24
2.2.3 Methods for structuring an assurance case	25
2.2.4 Depth and Breadth of Assurance Case Notation Research	28
2.3 Modelling language, metamodel, and visual notation	29
2.3.1 Modelling language and metamodel	29
2.3.2 Visual notation	31
2.3.2.1 Basic concepts	31
2.3.2.2 Defining visual notation effectiveness and efficiency	32
2.3.2.3 Visual notation design and evaluation methods	33
2.3.2.4 Visual notation evaluation studies	35
2.4 Visual inheritance in object-oriented paradigm	37
2.4.1 Object-oriented concept	37
2.4.2 Inheritance	38
2.4.3 Visual inheritance	39
2.5 User-centered design	40
2.5.1 User-centered design concepts	40

2.5.2	User-centred design approach	40
2.6	Summary	42
3	Visual notation design approach	44
3.1	Introduction	44
3.2	Overview	45
3.3	Step 1: Create a design path	49
3.3.1	Metamodel elements identification	50
3.3.2	Elements prioritisation	52
3.3.3	Design path creation	53
3.4	Step 2: Identify design constraints	55
3.4.1	Visual variables selection	56
3.4.2	Identify existing notations	60
3.4.3	Identify existing design principles	61
3.4.4	Identify notation usability	65
3.5	Step 3: Consider visual inheritance and create visual representation	71
3.6	Step 4: Document design rationale	83
3.7	Summary	86
4	The development of the SACM notation	87
4.1	Introduction	87
4.2	Design scope	87
4.3	User involvement	89
4.4	Design path	90
4.4.1	Elements identification	90
4.4.2	Element prioritisation	95
4.4.3	Design path creation	96
4.5	Design constraints	99
4.5.1	Selected visual variables	99
4.5.2	Considered existing notations	99
4.5.3	Considered notation design principles	100
4.5.4	Considered notation usability	101
4.6	SACM argumentation notation	102
4.6.1	'SACMElement'	102
4.6.2	'ModelElement'	105
4.6.3	'Assertion'	106
4.6.4	'Claim' and its variants	110
4.6.5	'AssertedRelationship' and its variants	118
4.6.6	'AssertedInference'	121

4.6.7	'ArtifactReference'	126
4.6.8	'AssertedEvidence'	129
4.6.9	'AssertedContext'	131
4.6.10	'AssertedArtifactSupport'	134
4.6.11	'AssertedArtifactContext'	136
4.6.12	'ArgumentReasoning'	138
4.6.13	'metaClaim'	139
4.6.14	'ArgumentPackage' and its variants	141
4.6.15	'ArgumentGroup'	145
4.7	Design rationale of the created visual representation	146
4.8	Summary of SACM notation	147
4.9	Application examples	148
5	Evaluation of the design approach	161
5.1	Introduction	161
5.2	Study definition	162
5.2.1	Objectives	162
5.2.2	Experimental objects	164
5.2.3	Perspective	164
5.3	Planning	164
5.3.1	Participant selection criteria	165
5.3.2	Context selection	165
5.3.3	Hypothesis formulation	165
5.3.4	Variable selections	166
5.3.5	Study design	166
5.3.6	Instrumentation	169
5.4	Operation	170
5.4.1	Preparation	170
5.4.2	Execution	170
5.4.3	Data validation	170
5.4.4	Data collection	171
5.5	Data analysis and interpretation	171
5.5.1	Demographics	171
5.5.2	Visual representation intuitiveness	172
5.5.3	Accurate use of the notation	174
5.5.4	Notation that is easy to use	180
5.5.5	Identification of element variants	183
5.5.6	Utilisation of visual variables	185
5.5.7	Participants' perception towards the notations	187

5.6	Threats to validity	188
5.7	Challenges and limitations	189
5.8	Summary and conclusion	190
6	Evaluation of the SACM notation	193
6.1	Introduction	193
6.2	Design decision evaluation	194
6.2.1	Study definition	194
6.2.2	Planning	198
6.2.3	Operation	201
6.2.4	Data analysis and interpretation	202
6.3	Experienced user evaluation	217
6.3.1	Study definition and planning	217
6.3.1.1	Objective	217
6.3.1.2	Survey design	217
6.3.1.3	Participants selection criteria	218
6.3.1.4	Pilot	218
6.3.2	Data analysis	218
6.3.2.1	Pre-processing	218
6.3.2.2	Analysis procedure	218
6.3.3	Survey Result	219
6.3.3.1	Demographics	219
6.3.3.2	Clarity of the notation	219
6.3.3.3	Usefulness of SACM features and their representation	224
6.3.4	Threats to validity	225
6.4	Evaluation through tool support	227
6.5	Evaluation by international standard body	228
6.6	Summary and conclusion	228
7	Conclusions	231
7.1	Thesis contributions	231
7.1.1	An approach to develop a visual notation based on a defined metamodel	231
7.1.2	A visual argumentation notation for the Structured Assurance Case Metamodel	232
7.1.3	Empirical evaluation of the SACM argumentation notation	233
7.2	Challenges and limitations	233
7.3	Future work	234
7.3.1	Generality of the proposed visual notation design approach	234
7.3.2	Refinement of the SACM notation	235

7.3.3	Experiment involving a larger number of experienced assurance case users with more complex examples	235
7.4	Overall conclusions	235
A	Visual representation identity of SACM notation	237
B	Empirical study data: SACM notation - GSN	284
C	Alternate SACM notation	308
D	Empirical study data: SACM notation - SACM alternate notation	310
	References	333

List of Tables

3.1	Visual representation identity template	84
3.2	Student visual representation identity	84
4.1	Extracted SACM argumentation metamodel elements	91
4.2	Identified elements to be visualised as part of the SACM notation design (First iteration)	94
4.3	Prioritised elements	96
4.4	Design path (tabular view)	98
5.1	Study design	167
5.2	Average score of the assurance case construction task using SACM notation and GSN	177
5.3	Assurance case model construction task result (SACM notation Vs GSN): group comparison	177
5.4	t-test results for the accurate use of the notation study	179
5.5	Timing data: average time spent by the participants from both Group A and B to create the correct assurance case models using SACM notation and GSN	181
5.6	t-test results for the effectiveness of the notation in terms of time required by the participants to create an assurance case	182
5.7	Summary of participants' perception from each group: SACM notation and GSN	188
5.8	Summary of participants' perception: SACM notation and GSN	188
6.1	Study design: SACM notation and its alternate notation	200
6.2	Average score of the assurance case construction task using SACM notation and SACM alternate notation	206
6.3	Assurance case model construction task result: group comparison	207
6.4	t-test results for the accurate use of the notation study (SACM Vs Alternate notation	208
6.5	Timing data: average time spent by Group A and B participants to create the correct assurance case models using SACM notation and SACM alternate notation	210
6.6	t-test results for the effectiveness the notation in terms of time required by the participants to create an assurance case using SACM notation and SACM alternate notation	211
6.7	Comparison of participants' responses related to the identification of 'Claim' element variants of SACM notation and Alternate SACM notation	214
6.8	Summary of participants' perception from each group: SACM notation and Alt. SACM notation	216

6.9	Summary of participants' perception: SACM notation and Alt. SACM notation . . .	217
6.10	Survey participants: demographics	219
6.11	The usefulness of SACM elements	225
6.12	Example of participants' feedback regarding the SACM visual representations	226

List of Figures

2.1	Problems of Textual Argument [1]	24
2.2	Example of GSN Visual Notations	25
2.3	Example of CAE Visual Notations	27
2.4	Metamodel example (adopted from [2])	31
2.5	Diagrammatic Communication [3]	32
2.6	Semiotic Quality	34
2.7	Bertin's Visual Variables [4]	35
2.8	Polymorphism example	38
2.9	(a) Single Inheritance; (b) Multiple Inheritance	39
2.10	Example of the visual inheritance in User Interface design	40
3.1	The visual notation life cycle	46
3.2	The proposed visual notation design approach	47
3.3	Visual notation design process	48
3.4	Step 1: Create a design path	49
3.5	Metamodel X	51
3.6	Example of a design path of the Metamodel X	55
3.7	Step 2: Identify design constraints	56
3.8	Example of Location/Position in Use Case diagram	57
3.9	Example of Size indicated by the thickness of the line used for Join	57
3.10	Example of DeMarco Data Flow Diagram visual representations	58
3.11	Example of Orientation in visual representation: a fragment of an argument structure adopted from [5]	58
3.12	Example of Brightness in visual representation	59
3.13	Example of Texture as used in UML Sequence diagram	60
3.14	Example of the same visual representation used to represent two different elements with different semantics	60
3.15	Example of the same visual representation used to represent two different elements with different semantics	60
3.16	Defensive State visual representation [6]	62
3.17	A Stick-man icon represents an Actor in UML	63
3.18	The UML Package	63

3.19	The SysML Package	63
3.20	I* Agent visual representation	64
3.21	Semantic transparency of the visual representation types	64
3.22	Visual distance between visual representations of De Marco DFDs Entity and Process	65
3.23	Redundant coding used in UCM Actor visual representation	65
3.24	Example of a complex visual representation (in context of a hand-drawn visual representation)	66
3.25	The original UML Package visual representation	67
3.26	The visual representation of Compromised State	68
3.27	Alternative Compromised State visual representation as suggested in [6]	68
3.28	The use of Colour as a syntax highlighting technique [7]	69
3.29	The use of Colour as part of the Opacity-Driven Graphical Highlight technique [8]	70
3.30	Example of a model that used Colours as a syntax highlighting technique but was printed using a black-and-white printer	71
3.31	A screenshot of the Artoo argumentation tool [5]	72
3.32	Step 3: Create visual representation	72
3.33	Example of Generalisation relationship	73
3.34	Example of a design path of the Student Metamodel	75
3.35	Example of the visual inheritance	76
3.36	Step 4: Design rationale documentation	83
4.1	SACM Metamodel from the SACM specification version 2.0 [9]	88
4.2	Notation user involvement in the development and evaluation of the SACM notation	90
4.3	SACM Argumentation Metamodel from the SACM specification version 2.0 [9]	91
4.4	Design path of the SACM argumentation notation	98
4.5	Considering UML notation in the design of the SACM notation: (a) the first version of the proposed ‘ArtifactReference’ visual representation; (b) the UML comment/note visual representation	100
4.6	Considering UML notation in the design of the SACM notation: Revised ‘ArtifactReference’ visual representation in order to avoid a visual representation clash with the existing notation	100
4.7	Citation visual representation	104
4.8	The use of dashed line to denote ‘isAbstract’	104
4.9	‘Claim’ visual representation	111
4.10	‘Claim’ statement example	112
4.11	Different types of ‘Claim’ in SACM	113
4.12	The application of element identifier (ID) in a ‘Claim’ visual representation	114
4.13	The example of ‘needsSupport’ when being applied in a Claim (i.e. ‘needsSupport Claim’)	114

4.14	The visual representation of an ‘assumed Claim’	115
4.15	The visual representation of an ‘axiomatic Claim’	115
4.16	The visual representation of an ‘defeated Claim’	116
4.17	The visual representation of the ‘asCited Claim’	117
4.18	‘abstract Claim’ visual representation	117
4.19	Types of ‘AssertedRelationship’	121
4.20	‘AssertedInference’ visual representation	122
4.21	Previous design of the ‘AssertedInference’ visual representation - without dot in the middle of the line	123
4.22	Types of ‘AsseretedInference’ relationship	124
4.23	Optional type of ‘abstract AssertedInference’ visual representation	125
4.24	Multiple type of ‘abstract AssertedInference’ visual representation	125
4.25	Choice type of ‘abstract AssertedInference’ visual representation	126
4.26	‘ArtifactReference’ visual representation	126
4.27	Previous version of the ‘ArtifactReference’ visual representation	127
4.28	‘abstract ArtifactReference’ visual representation	128
4.29	‘asCited ArtifactReference’ visual representation	129
4.30	‘AssertedEvidence’ visual representation	130
4.31	Previous version of the ‘AssertedEvidence’ visual representation	130
4.32	Types of ‘AssertedEvidence’ relationship	131
4.33	‘AssertedContext’ visual representation	132
4.34	Previous design of the ‘AssertedContext’ visual representation	133
4.35	Types of ‘AssertedContext’ relationship	134
4.36	‘AssertedArtifactSupport’ visual representation	134
4.37	Types of ‘AssertedArtifactSupport’ relationship	136
4.38	‘AssertedArtifactContext’ visual representation	136
4.39	Types of ‘AssertedArtifactContext’ relationship	138
4.40	‘ArgumentReasoning’ visual representation	138
4.41	‘metaClaim’ visual representation	140
4.42	SACM Package convention	142
4.43	‘ArgumentPackage’ visual representation	142
4.44	Types of ‘ArgumentPackage’	143
4.45	The ‘ArgumentGroup’ visual representation	145
4.46	Summary of SACM argumentation notation	147
4.47	Example of ‘Claims’ supporting another ‘Claim’. The ‘AssertedInference’ relationship is used to associates these elements.	148
4.48	Example of ‘ArtifactReference’ providing context to a ‘Claim’. The ‘AssertedContext’ relationship is used to associates these elements.	149

4.49	Example of ‘ArtifactReference’ providing context to the ‘AssertedInference’ relationship. The ‘AssertedContext’ relationship is used to associates these elements.	149
4.50	Example of ‘ArgumentReasoning’ providing additional description to the assurance case.	150
4.51	Example of a ‘Claim’ provides necessary context for another ‘Claim’. The ‘Asserted-Context’ relationship is used to associates these elements.	151
4.52	Example of an ‘ArtifactReference’ as evidence to support ‘Claims’. The ‘AssertedEvidence’ relationship is used to associates these elements.	151
4.53	Example of an ‘axiomatic Claim’	152
4.54	Example of an ‘asCited Claim’	153
4.55	Example of a ‘needsSupport Claim’	154
4.56	Example of dialectical assurance argument	154
4.57	Example of abstract elements in SACM notation	155
4.58	Another example of abstract elements in SACM notation	156
4.59	Example of ‘metaClaim’	157
4.60	Example of ‘ArgumentPackages’	158
4.61	Example of ‘ArgumentGroup’	159
5.1	The participant’s responses regarding the ‘ArtifactReference’ visual representation intuitiveness question	173
5.2	The participant’s responses regarding the ArgumentPackage visual representation intuitiveness	174
5.3	Tabular assurance case used as a case study in the assurance case model construction task	175
5.4	Data distribution Group A and B for the accurate use of the notation study (using (a) SACM notation and (b) GSN)	179
5.5	Performance results of Group A and B with respect to the times spent in creating the assurance case models using both notations (in hh:mm:ss)	180
5.6	Performance results of Group A with respect to the times spent in creating the assurance case models using both notations (in hh:mm:ss)	181
5.7	Performance results of Group B with respect to the times spent in creating the assurance case models using both notations (in hh:mm:ss)	182
5.8	Data distribution Group A and B for the effectiveness of the notation study - time required to create an assurance case (using (a) SACM notation and (b) GSN)	182
6.1	SACM Alternate notation (Example of ‘Claim’ and ‘AssertedRelationship’ type) . . .	197
6.2	SACM Alternate notation example	197
6.3	The participant’s responses regarding the intuitiveness of the ‘ArtifactReference’ visual representation	203

6.4	The participant's responses regarding the intuitiveness of the 'ArgumentPackage' visual representation	204
6.5	Data distribution Group A and B for the accurate use of the notation study (using (a) SACM notation and (b) SACM alternate notation)	208
6.6	Performance results of Group A and B with respect to the times spent in creating the assurance case models using both notations (in hh:mm:ss)	210
6.7	Performance results of Group A with respect to the times spent in creating the assurance case using SACM notation and SACM alternate notation(in hh:mm:ss)	210
6.8	Performance results of Group B with respect to the times spent in creating the assurance case using SACM notation and SACM alternate notation (in hh:mm:ss)	211
6.9	Data distribution Group A and B for the effectiveness of the notation study - time required to create an assurance case (using (a) SACM notation and (b) SACM alternate notation)	211
6.10	Assurance case A presented using GSN	220
6.11	Assurance case A presented using SACM notation	220
6.12	Assurance case B presented using GSN	221
6.13	Assurance case A presented using SACM notation	221
6.14	Implementation of the SACM notation in TouchDraw tool as one of its notation library	227
6.15	SACM 2.1 equipped by the proposed notation	228

Acknowledgements

This thesis would not have been possible without the guidance, advice, and encouragement from my supervisors, Richard Hawkins, Ibrahim Habli, and Tim Kelly (from 2016 to 2019). I am grateful to them for giving me the opportunity to work under their supervision with all their input, valuable ideas, and support.

I would also to thank Rob Alexander as my internal examiner for continually giving me valuable feedback during my PhD time. He always gives me interesting ideas and feedback to improve my work. Thank you also to Alan Wassyng as my external examiner for the precious feedback and discussion during the viva. I also thank the HISE members who help me a lot, especially in the brown bag session where I can share my progress, and their feedback helps me to explore further to improve my work.

I would like to thank my family, particularly my Mum and Dad, and also Ninan. Their constant support, encouragement, and love are my inspiration.

Finally, I thank my wife, Gia Wulandari, who also doing her PhD at the same time as me, for her love, support, and patient.

Declaration

Some of the material presented in this thesis has previously been published in the following publications or standard:

- Nungki Selviandro, Richard Hawkins, and Ibrahim Habli. "A visual Notation for the representation of assurance cases using SACM". In Seventh International Symposium on Model-Based Safety and Assessment (IMBSA 2020). Lisbon, Portugal. September 2020.
- Object Management Group (OMG). "Structured assurance case metamodel (SACM) Specification Version 2.1 - Annex C and D". USA. April 2020.
- Nungki Selviandro, Tim Kelly, and Richard David Hawkins. "The visual inheritance structure to support the design of visual notations." In Third International Workshop on Human Factors in Modeling (HuFaMo'18). Copenhagen, Denmark. October 2018.

Except where stated, all of the work contained within this thesis represents the original contribution of the author.

Chapter 1

Introduction

In this chapter, the motivation of this thesis is described, including the research background and the identified problems. The research questions, along with the research objective, and hypothesis, are also provided in this chapter in order to specify the aims of this thesis. Finally, in the last section, the thesis structure is outlined.

1.1 Motivation

An assurance case is commonly used to provide a clear and structured basis for analysing and communicating the assurance arguments and evidence regarding a particular system in a specific environment. An assurance case consists of a related set of auditable elements, such as claims, arguments, and evidence, used to demonstrate that a defined system will satisfy particular properties of interest such as safety or security requirements. In a complex system, structured arguments will typically be large and complicated; therefore the argument and evidence must be clearly documented to ensure they are communicated in a clear and defensible way between the different system stakeholders such as developers, reviewers, and regulators [9].

An assurance case can be represented using a textual (i.e. natural language) and/or graphical approach. The Goal Structuring Notation (GSN) and the Claim-Arguments-Evidence (CAE) are two examples of established assurance case frameworks that can be used to represent an assurance case using graphical representations [10]. These two frameworks have been widely adopted in various domains to represent assurance cases, and examples related to their use can be accessed in the literature, for example, in [1, 11–13].

The Structured Assurance Case Metamodel (SACM) is a specification that defines a metamodel for representing structured assurance cases. It is issued and published by the Object Management Group (OMG) to improve standardisation and interoperability in assurance case development [9]. SACM is

built upon the existing and established assurance case frameworks such as GSN and CAE. It provides a richer set of features, in terms of expressiveness, than existing assurance case frameworks. SACM was also developed to better support a model-based approach to assurance case development. The potential advantages of using SACM for creating assurance cases have been discussed in [14]. Despite the advantages of adopting SACM for creating assurance cases, one of the limitations of the SACM is related to the lack of visual representation that can be used to visually represent the assurance cases. Unlike the GSN and CAE, SACM is not equipped with any visual notation. Although SACM models could be represented visually through the transformation of the models to a different argumentation notation such as GSN or CAE, these other notations do not provide the richness of SACM, and the transformations can be complicated and incomplete. It is therefore highly desirable to provide the ability to visually represent SACM models directly.

A visual notation can be seen as the interface of the modelling language and the users. It plays an essential role in communicating complex information, especially when communicating with non-technical users. A visual notation can also be seen as an alternative representation to the textual form [3, 15]. Similar to GSN and CAE, but from another domain, the Unified Modeling Language (UML), Business Process Model and Notation (BPMN), and Systems Modeling Language (SysML) are examples of modelling languages that also provide users with a visual notation.

In the visual modelling language development process, the semantic constructs and the language rules can be defined in a metamodel by the language developers together with the end-users, who help to refine the language concepts [16]. The visual notation should be able to be developed based on the defined metamodel as an alternative representational form to the textual representation of the language. However, a limited approach is available that can be adopted to develop a visual notation that considers the defined metamodel as part of the process. Visual modelling languages can also be found whose development is not based on a metamodel. In this context, those visual modelling languages were invented before the concept of the metamodelling approach was introduced. An example of this is the I* modelling language. It was developed before the concept of metamodelling approach was introduced as part of the development of a modelling language. After I* was widely adopted by its users in the requirement engineering domain and the concept of metamodelling approach was introduced, there were several proposals that attempted to propose a metamodel for the I*, such as [17, 18]; meanwhile the visual notation of I* was already widely adopted by its users. In contrast to I*, the SACM is developed and specified in a metamodel. However, there is no visual notation provided in the specification of the SACM 2.0.

The structured argument is the heart of any assurance cases. Part of the SACM specification defines a metamodel for representing structured arguments. This part of the SACM is one of the main parts that can be used to develop assurance cases [9]. This thesis aims to provide a visual notation for the SACM argumentation because there is no visual notation provided for this specification (SACM 2.0). Since the concept for representing structured assurance cases in SACM is specified using a metamodel, the

metamodel and its structure should therefore be considered in the development process of the SACM visual notation.

The visual notation design process is a crucial factor in determining the effectiveness of the notation in terms of accurately and efficiently conveying the information represented in the model [19]. There are several visual notation design approaches that exist to support the visual notation designer to develop a visual notation such as [3, 20, 21]. One of the most cited approaches in the domain of visual notation design and evaluation is known as the Physics of the notation (PoN). This approach aims to ensure that a visual notation is designed to be cognitively effective by providing nine visual notation design principles that can also be used as variables to evaluate a visual notation [3]. However, this approach received some critiques in the literature, mainly focused on the difficulty of systematically applying the provided principles. Although there are several proposals that attempted to enhance the PoN [22–24], there is still a gap regarding how to apply the PoN principles systematically. Also, there is a lack of empirical evidence to prove that the proposed improvements are sufficient and can be effectively adopted to develop a visual notation [15]. Moreover, there is no systematic guideline provided in PoN principles as well as other existing visual notation design approaches that specifically provide visual notation design steps based on a defined metamodel.

Since there is a limitation regarding the visual notation design approach that can be adopted to develop the SACM notation, this thesis aims to propose a visual notation design approach that considers the structure of a defined metamodel as part of the process. By providing such approach, the lack of systematical approach in visual notation design, specifically, in the context of the development of a visual notation that consider the defined metamodel, can be addressed. The proposed visual notation design approach is used to develop the SACM notation. This notation is developed to address the current limitation of the SACM 2.0 specification where there is no visual notation provided as an alternative to the textual representation.

1.2 Research questions

Based on the above discussions related to the motivation of this thesis, the following research question is defined to help specify the work that will be presented in this thesis:

1. How can an effective SACM visual notation be developed?

Due to the limitation of the existing visual notation design approaches that can be adopted to develop the SACM notation, in order to answer this question, this thesis will address the following research questions:

2. How can a visual notation be developed based on a defined metamodel?

The following additional research question that is related to the evaluation of the thesis is also considered:

3. How effective is the resulting notation that is designed using the proposed visual notation design approach?

The definition of an effective notation used in this thesis is discussed further in Chapter 3.

1.3 Thesis objectives and hypothesis

As described in the motivation of this thesis, the objective of this thesis is to develop a visual notation design approach that can be used to develop a visual argumentation notation for SACM. The proposed visual notation design approach needs to consider the structure of a defined metamodel due to reducing the current limitation in visual notation design studies where there is a lack of a systematical approach that can be adopted to develop a visual notation that considers the defined metamodel.

Based on the research objectives, this thesis addresses the following hypothesis:

Through the application of the proposed visual notation approach, an effective SACM argumentation notation can be developed.

1.4 Thesis structure

This thesis is organised into seven chapters:

- **Chapter 2** presents the literature review relating to existing assurance case notations, the SACM specification, modelling languages and metamodels, visual notation design approaches, and the user-centred design approach.
- **Chapter 3** presents the proposed visual notation design approach. This approach focuses on the development of a visual notation based on a defined metamodel structure. This approach also considers multiple aspects such as existing visual notation design approaches, existing notation in the domain, and the utilisation of visual variables.
- **Chapter 4** shows the application of the proposed visual notation design approach to the development of a visual argumentation notation for SACM.
- **Chapter 5** presents the evaluation of the proposed visual notation design approach through an empirical study. This compares the SACM argumentation notation developed by following the proposed approach with an existing assurance case notation developed by another notation design approach.

- **Chapter 6** presents the evaluation of the SACM argumentation notation. The evaluation consists of several parts such as expert-based evaluation (involving experienced assurance case users such as assurance case developers, specialists, and project managers), an empirical study involving novice participants (participants without theoretical or practical knowledge of assurance case notations), and tool support implementation.
- **Chapter 7** presents a summary of the thesis contributions, the limitations, and the suggested directions of future research.

Chapter 2

Literature Survey

2.1 Introduction

This chapter presents a review of the published literature as the context for this research. As stated in the previous chapter, the objective of this thesis is to propose a visual notation design approach that can be used to develop a visual notation for the SACM argumentation specification. To support this objective, we need to study existing literature from different knowledge areas. We outlined four major contributing fields of knowledge that we describe in this chapter:

- Structured assurance case.
- Modelling language, metamodel, and visual notation.
- Visual inheritance in object-oriented paradigm.
- User-centered design

The details of our literature review result related to these knowledge areas is described in the following sections.

2.2 Structured assurance case

The concept of an assurance case is that it provides a framework for analysing and communicating the assurance arguments and evidence about a particular system in a specific environment (e.g. covering safety requirements of the systems) [9]. In this section, we discuss the definition of the assurance case and followed by approaches and methods in representing and structuring assurance cases.

2.2.1 Structure assurance case definition

In [25], an assurance case is defined as:

"... a reasoned and compelling argument, supported by a body of evidence, that a system, service or organisation will operate as intended for a defined application in a defined environment."

Based on the definition above, arguments and evidence form the foundation of an assurance case. Therefore, both argument and evidence need to be expressed clearly in the assurance case document.

An assurance case document facilitates information exchange between various system stakeholder such as developers, acquirers, and regulators, where the knowledge related to the system is communicated in a clear and defensible way (e.g. about the safety and security aspect of a system). Each assurance case should communicate the scope of the system, provides the operational context, the claims, the safety and/or security arguments, along with the supporting evidence [9].

In order that an assurance case can be developed, reviewed, and presented, it is important that the assurance case be documented clearly. Sometimes an assurance case can be very large and complicated for example in complex systems. Therefore, the documented argument of the assurance case should be structured to be comprehensible to all stakeholders. It should also be clear how the evidence is being asserted to support the arguments [25].

2.2.2 Assurance case representations

There are several ways that can be used to represent an assurance case such as use free text approach (natural language) [26], tabular structures, or graphical-based notation [27]. However as noted in [27] the use of free-text approach may pose difficulties for the reader in identifying the individual elements and structure of the argument. An example can be seen in Figure 2.1. This type of textual safety argument with multiple cross-references can be awkward and disrupt the flow of the main argument [27].

<p>For hazards associated with warnings, the assumptions of [7] Section 3.4 associated with the requirement to present a warning when no equipment failure has occurred are carried forward. In particular, with respect to hazard 17 in section 5.7 [4] that for test operation, operating limits will need to be introduced to protect against the hazard, whilst further data is gathered to determine the extent of the problem.</p>
--

FIGURE 2.1: Problems of Textual Argument [1]

Similar with free-text format, although tabular approach offers a simple means of structuring an argument, when presenting safety arguments using this approach we might lose the clarity and the flow of the argument due to the limitation in expressing relationship and flow of the argument using tables [27].

Another way is use graphical-based approach. Adopting this approach is believed could help to ease the difficulties in writing and certifying the safety cases, since the structure of the argument and the link to the evidence can be easily identified [26, 28].

When adopting graphical-based approach for representing safety cases, at least, there are two widely adopted notations that can be used: Goal Structuring Notation (GSN) [27] and Claim-Argument-Evidence (CAE) [29], where both of them were influenced by Toulmin's concept of argument pattern [26, 30]. The difference between them lies in notations that can be used to visually represent the argument structure.

2.2.3 Methods for structuring an assurance case

There are several methods that can be used to structure an assurance case. In the previous subsection we have mentioned GSN and CAE where both of them provide graphical notations to represent assurance cases. Other than GSN and CAE, there is another method, called as SACM argumentation [9], that can be used to structure an assurance case. However, this method currently is not equipped with any graphical or visual notations. As follows we summaries the concept of GSN, CAE, and SACM argumentation:

2.2.3.1 GSN

According to GSN community standard in [25], *GSN is a graphical argument notation which can be used to document explicitly the elements and structure of an argument and the argument's relationship to evidence*. It adopts a hierarchical paradigm where the top claim is on the top of the diagram and is followed by other GSN elements such as 'strategy', 'sub-goal', and 'solution'.

GSN provides guideline in how to construct an assurance case using GSN notations, which can be found in GSN Community Standard. Figure 2.2 illustrates the example of GSN Visual Notations.

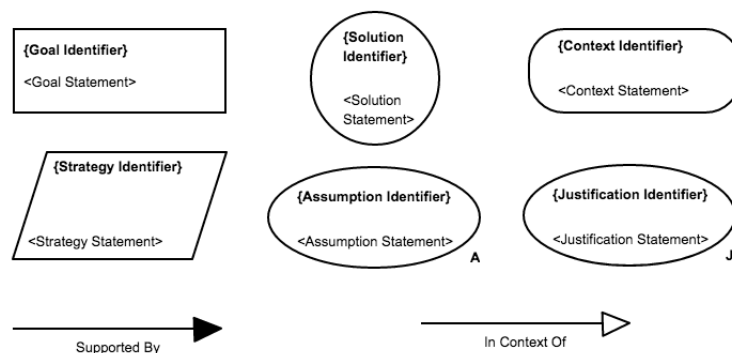


FIGURE 2.2: Example of GSN Visual Notations

The following are examples of the GSN elements:

- A *Goal*, presents a claim forming part of the argument. It is rendered as rectangle and must be expressed as a proposition in the form of a noun-phrase + verb-phrase sentence.
- A *Strategy*, describes the nature of the inference that exists between a goal and its supporting goal(s). It is rendered as parallelogram and must contain a brief description of the argument approach.
- A *Solution*, presents a reference to an evidence item(s). It is rendered as a circle and must be stated as noun-phrases.
- A *Context*, presents contextual information or a statement. It is rendered as rectangle with round corner and must be expressed as a noun-phrase for reference to an artifact or for explanatory contextual information.
- An *Assumptions* and a *Justifications*, provide additional information necessary for the correct understanding of the argument. Both of them are rendered as an oval with the letter 'A' at the bottom-right for the assumption and 'J' for the justification.
- *SupportedBy* relationship element, it declares an inferential or evidential relationship. It is rendered as a line with a solid arrowhead.
- *InContextOf* relationship element, it declares a contextual relationship. It is rendered as a line with a hollow arrowhead.

The elements of the GSN can be linked together in a network and described as a 'goal structure'. Other than that it can be used to representing an assurance case, GSN was developed to facilitates specific purpose such as support modular safety case development and support mechanisms to represent argument patterns as a way of explicitly capturing and documenting reusable assurance case elements. Further explanation and complete documentation of the GSN elements can be found in [25]. Literature that discusses the application of the GSN, for example, can be found in [31–33].

2.2.3.2 CAE

CAE is also a method for structuring an assurance case that adopts a graphical approach. However, it has different argument elements and notations compared with GSN. The argument notation elements of CAE from [29, 34–36] can be summarised as follows:

1. *Claim (sub-claims)*, a statement asserted within the argument that can be assessed to be true or false.
2. *Argument*, a description of the argument approach presented in support of a claim.
3. *Evidence*, a reference to the evidence being presented in support of the claim or argument.

4. *Side-Warrant*, a statement about the reason behind why we can deduce the top-level claim from the sub-claims and under what circumstances the argument is valid.
5. *Supports*, relationship-type element. It explains relation between 'argument' and 'claim'.
6. *Is a sub-claim of*, relationship-type element. It explains relation between 'sub-claim' and 'argument'.
7. *Is evidence for*, relationship-type element. It explains relation between 'evidence' and 'sub-claim'.

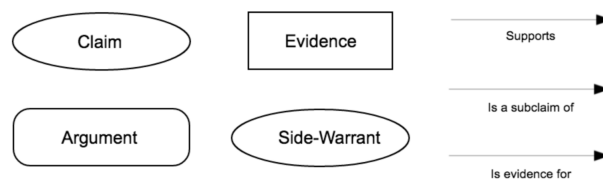


FIGURE 2.3: Example of CAE Visual Notations

Figure 2.3 shows the CAE visual notations. In general, both CAE and GSN concept are quite similar since both of them were influenced by Toulmin's concept of structured argument pattern for systematically structuring the argument. However, GSN provides more specific argument elements for explaining specific context and additional information such as *Context*, *Justification*, and *Assumption*. Literature that discuss the application of CAE can be found, for example, in [30, 36].

2.2.3.3 SACM Argumentation

According to SACM documentation in [9], SACM is a specification that defines a metamodel for representing structured assurance cases. It is published by The Object Management Group (OMG) to improve standardisation and interoperability for assurance cases. SACM consists of several parts that are related to the assurance case technical specifications. Part of the SACM specification defines a metamodel for representing structured arguments.

In SACM, structured arguments are represented explicitly by the Claims, citation of artifacts or ArtifactReferences (e.g. Evidence and Context for claims), and the relationships between these elements [9]. The following relationships are defined:

- AssertedInference for relationship between Claims
- AssertedEvidence for ArtifactReference (as Evidence) and Claims
- AssertedContext for ArtifactReference (as Context) and Claims
- AssertedArtifactSupport for ArtifactReference (as Evidence) supporting ArtifactReference (as Evidence and Context)

- AssertedArtifactContext for ArtifactReference (as Context) supporting ArtifactReference (as Evidence and Context)

In addition to these core elements, it is also possible to provide:

- Description of the ArgumentReasoning associated with AssertedInference or AssertedEvidence
- Counter-Argument and Counter-Evidence (via isCounter: Boolean)
- Modular structured argument (via ArgumentPackage) including the mechanism to organise a specific selection of the ArgumentElements contained within the package (via ArgumentPackageInterface), and a mechanism to bound two or more argument packages (via ArgumentPackageBinding)
- An association of a number of ArgumentElements to a common group (via ArgumentGroup)

As mentioned before, the current version of the SACM argumentation specification is not equipped with any visual notation. We believed that the visual notation is an integral part of modern modelling languages and can help users in communicating their intended message in the form of a model (or models) to the reader [3]. It would therefore be beneficial if we provide a visual notation for the SACM argumentation metamodel. The SACM specification is described further in Chapter 4 in this thesis as its visual notation is developed using the proposed visual notation design approach.

2.2.4 Depth and Breadth of Assurance Case Notation Research

In order to capture the current state of assurance case notation research, We summarised related publications that are available and specifically observe the visual notation aspect of the assurance case. For example, we observed that there are at least four notation extension proposals from current assurance case visual notations, that are:

- D* framework as an extension for GSN as discussed in [37, 38] where the authors introduced the use of argument elements such as 'Actor' (as a node element), 'Depend On' and 'Belongs To' (as relationship elements). They argued that the used of these argument elements could enhance the users' comprehension aspect when implementing them to represent safety arguments for dependability systems.
- An extension for GSN for dealing with changes in assurance cases also can be found in [39]. The authors proposed supplemental notations for GSN such as 'Accepted Goal', 'Rejected Goal', 'Stated Goal', 'Applicable Strategy', and 'Not Applicable Strategy' where they claimed that these supplemental notations could facilitates comprehension in understanding assurance cases.

- Another extension for GSN also discussed in [40]. They introduced notations such as 'Or' and 'And' in order to facilitate goal decomposition analysis in safety engineering. They argued that these notations could be very useful in order to make a selection for requirements analysts to evaluate the available sub-goals.
- A concept of 'Multi-View' safety cases has been proposed by Flood in [41]. This concept developed to support GSN visual notations. The author argued that 'Multi-View' safety cases have the potential to filter information of interest to stakeholders, thus reducing complexity and increasing comprehension of the safety argument.

Based on the above previous work, we may imply that some of the users' needs in regards to the current safety case visual notations are not fully satisfied, hence they proposed notation extensions to fulfill their specific needs.

In addition to several work about notation extensions, there is also previous work that is concerned with comprehension aspect of assurance case notations. For example, Graydon in [42] reported his analysis about 'Context' argument element in GSN. He argued that the semantic aspect of 'Context' element should be defined in a more precise way, since he believed currently the definition of 'Context' element in GSN is ambiguous and could affect the users' comprehension when using the notation. Furthermore, Graydon's opinion about this aspect is also echoed by Rushby et al as noted in [26].

Matsuno and Yamamoto in [28] also noticed that the effectiveness aspect of the assurance case notations could be improved by providing more formal definition for the syntax. However in their publication, they do not specifically explain about what they mean by formal definition.

Previous work about the comprehension aspect indicate that further work is required to provide an improvement in the comprehension aspect of an assurance case visual notation in terms of formal definition. However, formal definition itself is separated from the visual representation aspect. Visual representation itself plays another important role to deliver a clear and comprehensive message to the stakeholders.

2.3 Modelling language, metamodel, and visual notation

2.3.1 Modelling language and metamodel

Models can be defined as explicit representations of some portions of reality as perceived by some actor. When a model is presented, it must follow a particular rule of construction and representation defined by a particular language [43, 44]. A model can be represented in a textual or graphical form. Textual representation encodes information using sequences of characters. Visual representation encodes information using spatial arrangements of graphic (and textual) elements. An example

of graphical/visual modelling language is UML, and as a corresponding textual modelling language specifically for the UML activity diagram is the Surface Language as presented in [45].

Due to the increasing needs to adopt modelling languages in a specific domain, many domain-specific languages have been created for a specific purpose. Metamodelling techniques are adopted to support the interaction, standardisation, interoperability between languages, and modelling tool implementation.

Metamodeling is the process of defining modelling languages where the common process is first to define the syntax of the language and then to describe its semantic [46]. The syntax of the language consists of abstract and concrete syntax [3, 47]. For graphical modelling languages, the concrete syntax is the graphical/visual notation. After the abstract and concrete syntax is created, they should be able to be mapped to each other. The result of the metamodelling process is the metamodel. The creation of the notation for a graphical modelling language is one of the steps to define the concrete syntax of a language.

A metamodel can be seen as a model that consists of statements about models. Therefore, a metamodel is also a model, but its universe of discourse is a set of models, e.g., models of interest to the metamodel's creator. A metamodel typically consists of several elements. For example, a metamodel that is presented using the UML approach might consist of different types of elements such as:

- Class

A class in UML defines either an abstract or concrete class. An abstract class is a class that cannot be instantiated in the model level. A concrete class is a class that can be instantiated in the model level. The instantiation of the concrete class can be in the form of text, graphical (visual), or a combination of both of them.

- Attribute

A class in a metamodel may have attributes. An attribute can define, for example, a state of a class based on a particular defined input. An attribute may be grouped by its visibility status such as private or public that define the accessibility status of the attribute.

- Relationship

Relationship in UML can be specified into several types such as association, aggregation, and composition. For example, an association can be used as a relationship between classes, which is used to show that the class instances could be linked to each other.

Metamodels have become popular along with the increasing demands of the adoption of the modelling language. The current specification of UML supports metamodeling techniques to extend the

capabilities of a language and to adapt it to specific modelling domains. An example of a tool that supports metamodeling is MetaEdit+¹.

Figure 2.4 illustrates an example of a metamodel. It is a metamodel of the basic notation for a language called TOSCA (The Topology and Orchestration Specification for Cloud Applications). TOSCA provides a well-defined way to model composite applications and to provide plans for automating their management [48]. As shown in Figure 2.4, the metamodel of the basic notation for TOSCA is presented using the UML approach with elements such as Class, Attribute, Association, and Inheritance relationship.

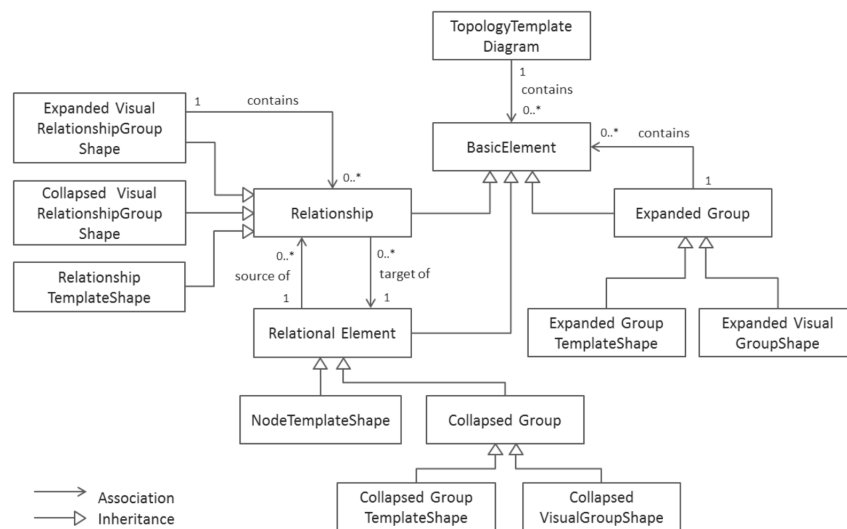


FIGURE 2.4: Metamodel example (adopted from [2])

In the next section, the visual notation as a form of the representation of the concrete syntax of a language is discussed.

2.3.2 Visual notation

2.3.2.1 Basic concepts

By definition, the term *notation* originally stems from the Latin verb 'Notare' and refers to a set of symbols. Graphical notation systems contain a set of graphical elements (pictograms), which can be combined with each other according to a set of rules [49].

Anatomically, a visual notation consists of a set of graphical symbols (visual vocabulary) that are equipped by a set of compositional rules (visual grammar) and the definition of each symbol (visual semantics). The visual vocabulary and visual grammar together form the visual (or concrete) syntax, meanwhile the graphical symbols are used to symbolise the semantic constructs that typically are defined by a meta-model, and the meanings of graphical symbols are defined by mapping them to the constructs they represent [3].

¹<https://www.metacase.com/mep/>

Visual notations form an integral part of the language of computer science and software engineering. Many visual notations have been developed to facilitate communicating complex information between technical and non-technical users. They are also used in different stages of software engineering, from requirements engineering through to maintenance. Visual notations are also adopted by practitioners in industry for strategic planning in the design of software systems. In industry, visual notations play a critical role in communicating with internal and external stakeholders [50]. This encourages visual notation researchers to conduct studies in how to maximise the effectiveness of visual notations.

2.3.2.2 Defining visual notation effectiveness and efficiency

Research related to visual notation design is an evolving research area. Many works have been published that attempt to design, evaluate and improve the design of the visual notations such as in [3, 6, 15, 17, 19, 20, 23, 51, 52]. It is important to have variables that can be used as a measurement to evaluate the design of the visual notation. In this case, effectiveness and efficiency are variables that are frequently used in visual notation design research as a measurement to evaluate a visual notation.

Visual notations in modeling languages are used as a form of communication between stakeholders that typically draws in diagrams. As shown in Figure 2.5, the information or message that needs to be communicated in the form of a diagram is encoded by the model developer or diagram creator, in this case as an example is X, and will be decoded by the reader Y. The diagram is encoded using a visual notation, which defines a set of conventions that both developer and reader understand. The effectiveness of communication between the developer and the reader is measured by the match between the intended message and the received message [3].

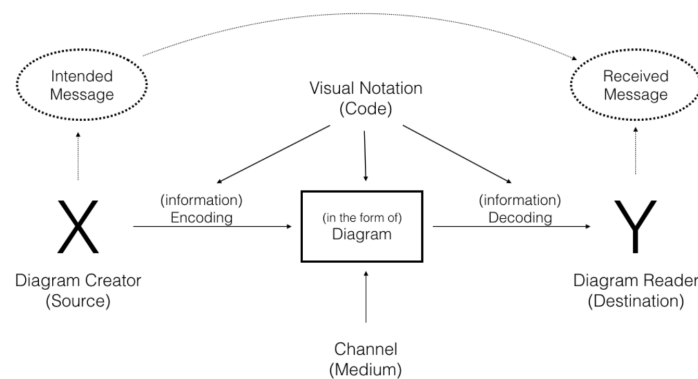


FIGURE 2.5: Diagrammatic Communication [3]

In [3], cognitive effectiveness of a visual notation is defined as the speed, ease, and accuracy with which a representation can be processed by the human mind. This definition is frequently used in visual notation design and evaluation research such as in [48, 53, 54]. There are numbers of visual notation empirical studies that cite this definition and they categorised accuracy as a variable to measure effectiveness and speed (response time in answering the question related to the notations)

as a variable to measure efficiency of a visual notation. For example in [54], the visual notation effectiveness is measured by the accuracy of users' correct interpretation regarding specific visual representation (symbol) of the visual notation and the visual notation used in a diagram. Besides that, they also measure the speed of reading and it is a measure of the response times for users to answer questions related to visual notation in a diagram.

2.3.2.3 Visual notation design and evaluation methods

Research in notation design is closely associated with research in conceptual modelling languages, specifically in evaluating the quality of the conceptual model [19]. Several theoretical frameworks have been proposed to evaluate and improve the quality of conceptual models, and some of them, such as Semiotic Quality (Sequal) [43], partially paid attention to the visual notation aspect. However, they were not designed exclusively to focus on the visual notation.

Sequal is a framework for defining the quality of a conceptual model. So, it cannot be used directly for assessing the effectiveness of a visual notation. The authors of this framework argue that Sequal is influenced by linguistic concepts of syntax, semantic, and pragmatic. Therefore, this framework focuses on these concepts to define the quality of syntax, semantics, and pragmatics of a conceptual model, and defines four aspects of modeling:

1. Language (L), consists of statements which has syntax and semantics. Syntax divided into alphabet (notations/symbols/vocabulary/constructs) and grammar (rules that define how to legally combine modeling constructs). Meanwhile, semantic divided into semantic (or meaning) for each construct and semantic for the statements.
2. Domain (D), consists of all possible statements that would be correct and relevant for solving the problem.
3. Model (M), set of statements that are actually made
4. Audience Interpretation (I), the set of statements that the audience thinks the model contains.

The primary sources for this framework are defined using the relationships between the model (M) and the three other sets (as illustrated in Figure 2.6).

Unfortunately, this framework can not be used for assessing the effectiveness of visual notations because this framework is developed by the author to be used for defining the quality of a conceptual model [55] [56]. However this framework give us some senses about the relations between model that could be visualised using visual notations with domain therefore we can define the semantic quality. Beside that we also can have insight about how to define syntatic quality in the context of this framework that measured by understanding the relationship between model and language where the language syntax and grammar are defined.

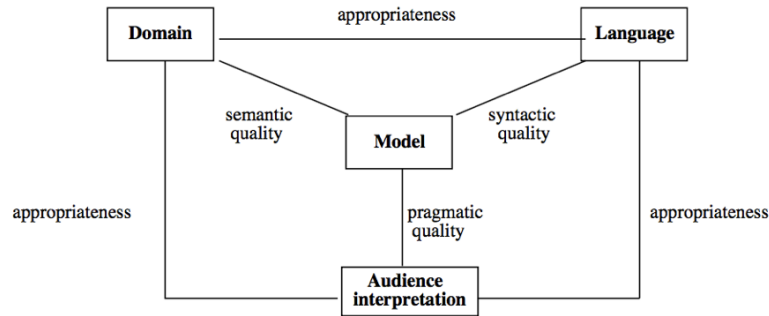


FIGURE 2.6: Semiotic Quality

Recently, a theory that focuses on visual notation design has been proposed, which is called The Physics of Notations (PoN). This theory aimed to ensure that visual notations are designed to be cognitively effective (i.e. the speed, ease, and accuracy with which a representation can be processed by the human mind) [3]. PoN provides nine principles that can be used either to design or to evaluate the visual notation. The followings are the overview of the PoN principles:

1. Semiotic Clarity: there should be a one-to-one correspondence between elements of the language (semantic constructs) and graphical symbols.
2. Perceptual Discriminability: refers to the ease and accuracy with which symbols can be differentiated from each other.
3. Semantic Transparency: refers to the use of graphical representations whose appearance suggests their meaning.
4. Visual Expressiveness: defined by the number of different visual variables used and the range of values (capacity) used for each variable.
5. Complexity Management: refers to the ability to present large amounts of information without overloading the human mind.
6. Cognitive Integration: this principle provides mechanisms to support the integration of information from different diagrams (only applies when multiple diagrams are used to represent a problem situation).
7. Dual Coding: the use of text to complement graphics.
8. Graphic Economy: the number of different graphical symbols should be cognitively manageable.
9. Cognitive Fit: use different visual dialects for different tasks and users when required.

There are numbers of existing notations that has been evaluated using the PoN principles such as UML

[57], UML Statechart notations [6], Misuse Case notations [53], I* [17], SEAM [50], UCM [58], and BPMN [59]. There are also several new notations that are designed using the PoN principles, such as VTML (The Visual Traceability Modelling Language) [60] and Larman's Operation Contracts [61]. However, this framework has received some criticism in the literature, mainly focused on the validation and difficulty in applying the principles since PoN does not prescribe any systematic method or process for the implementation of the principles. Several works, such as in [15, 22, 23, 62], have been proposed to address this issue.

Besides PoN, the Cognitive Dimensions (CDs) theory is also frequently discussed in notation design literature. CDs framework was introduced by Green, who emphasised the use of this framework as a "discussion tool" to aid non-HCI specialists in evaluating the usability of notational systems and information artifacts [63]. To do this, the CDs provides a vocabulary that can be used by designers when investigating the cognitive implications of their design decisions. Therefore, it does not explicitly provide any guideline or procedure for designing or evaluating a visual notation [20].

With the advancement of meta-modelling standards, the semantic constructs and grammatical rules of modelling languages tend to be defined in the metamodel to facilitate communication and tool support [17]. The notation should be defined afterwards referring to the defined metamodel. In order to design the visual syntax of the visual notation, we can use Bertin's visual variables theory.

Bertin defined eight visual variables that can be used to graphically encode information: brightness, orientation, size, shape, color, texture, horizontal and vertical positions[4]. Figure 2.7 shows Bertin's visual variables that can be used by notation designers to develop numbers of graphical symbols by combining the variables together in different ways. However, there is still lack of guideline that specifically can be used to design visual notation from a predefined metamodel.

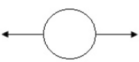
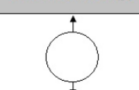




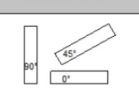

PLANAR VARIABLES		RETINAL VARIABLES		
Horizontal Position (x)		Shape	Color	Size
Vertical Position (y)				
		Brightness	Orientation	Texture
				

FIGURE 2.7: Bertin's Visual Variables [4]

2.3.2.4 Visual notation evaluation studies

Beside the above visual notation theories, we also need to understand how visual notation researchers evaluate a visual notation. There are several studies that have been conducted related to visual notation evaluation. The evaluation process can be categorised into two groups: expert-based evaluation and user-based evaluation.

Expert-based evaluation mostly adopt a method such as PoN for evaluation guideline. For example, study about I* notations effectiveness as reported in [17]. The authors analysed the I* notations use nine PoN principles and they found several problems such as symbol redundancy, symbol overload, and graphical complexity in I* notations. However, this study did not provide any validation in regards to their analysis results.

On the other hand, user-based evaluation is other way to evaluate an existing visual notation is involving the users of the notation in the evaluation process. Several studies have used this approach to assess existing visual notations. For instance studies that are conducted by Arning and Ziefle on the analysis of comprehensibility of C3 notations. C3 is a process modeling (PM) notation that is largely influenced by UML notations. As reported in [49], they conducted the naming method experiment where the participants of this study had to find the most adequate verbal term for the pictogram of the C3 notation system. The experiments involving different types of participants:

- novices, participants without theoretical or practical knowledge of PM at all
- general PM-experience, participants with theoretical or practical knowledge about PM methods
- specific C3-experience, respondents with theoretical or practical knowledge of the C3-method.

The result of this experiment shows that overall participants gave 52.5% correct replies. If we see from types of participants, only 39.8% of novice users gave correct replies while 53.8% of participants with general PM experience and 79.2% of participants with C3 experience gave correct replies.

Based on the result, the authors of this experiment stated that there was a low comprehension rate of C3 notation elements (across all users), which is on average far below the recommended recognition rate of more than 66% in ISO design guidelines (ISO 9186). The ISO standard for testing graphical symbols defined the minimum hit rates of 67% as the hit rate is required for acceptance of public information and safety symbols. The result indicates that C3 notation elements are not intuitively comprehensible for all users, but rather rely on specific previous knowledge. This experiment informed us that asking the users of the notations could give important feedback in order to understand the current comprehensibility aspect of a particular notation. Beside that the feedback can be very useful to design an improvement of the visual notation.

Beside those two examples, there also a study that compare the results from expert-based and user-based analysis of a visual notation [51]. This study concentrates on understanding the semantic transparency of UML notation by comparing the semantic transparency between the original UML visual notation, UML PoN-based visual notation (expert-based analysis), and UML user-based design visual notation by involving the users in the experiment process.

To measure graphical notations comprehensibility, the researchers use the hit rates or percentage of correct responses. The ISO standard for testing graphical symbols defined the minimum hit rates

of 67% as the hit rate is required for acceptance of public information and safety symbols. The result of their experiments shows that user-based symbols resulting mean of 71.5% hit rates which is above ISO minimum standard, higher than UML standard notations (22%), and PoN-based notations (33%). Therefore, this experiment inform us that involving users of the notation in design process can provide better semantic transparency aspect rather than designing the notation only based on expert analysis.

Based on three examples of visual notation effectiveness studies that are explained above, we conclude that other than involving expert, involving users is also important in the evaluation process of a visual notation.

2.4 Visual inheritance in object-oriented paradigm

In this section, we describe the concept of visual inheritance in the object-oriented paradigm and how this concept influences the basic idea in defining the design principles in this study.

2.4.1 Object-oriented concept

The object-oriented (OO) methodology is an approach in the system development that encourage the software developers to re-use the software components. This concept can be applied in the design phase, analysis, implementation, and maintenance.

The basic building block of the object-oriented concept is the object itself. An object in an OO system can represent organisational things (e.g. companies or departments), conceptual things (e.g. purchase orders or reservations), or physical things (e.g. managers or customers). Objects can also be a feature of computer implementation such as Graphical User Interface (GUI) frame [64].

In OO, an object is defined in terms of its class. A class also can be seen as a template for all objects of the class. An object that belongs to a certain class also known as an instance of that class. Therefore, the term of object and instance in OO are used interchangeably. A class typically have specific attributes and operations. All instances of the same class will have the same characteristics both their attributes and operations. The process of creating an instance of a class also known as instantiation [65].

The attributes of a class are the data items that define the class, for example, name, date of birth, and departments might be attributes of a class Student. On the other hand, the operations of a class combine to form the class behavior. The term operations of a class are closely related to term methods. Both terms refer to the procedures that form part of a class definition [64]. Attributes and operations of a class can be defined with a specific visibility method, for example as public or private [65].

All information related to an object (e.g. attributes and methods) are encapsulated in the object. The concept of *encapsulation* means that the only way to affect the object is to perform operations on that object and the user of that object can only know the operations that are available without knowing the details (e.g. implementation) of the operations. Encapsulation provides a mechanism, called as *data or information hiding*, to protect the data as the data can only be manipulated by the operations defined in the object. Meanwhile, the procedure of representing only important information related to an object without including the details is also known as *abstraction* [64–66].

It is possible in OO that for the same methods to act differently in different class. This concept is called as polymorphism that provides the ability to take on more than one form [65, 67]. For an example of this concept, there is a 'Person' class which has an attribute defined as 'name' and a method defined as 'talk', as in Figure 2.8. There are two instances of the 'Person' class: 'Baby' and 'Teenager' class. As stated before, all instances of the same class will have the same characteristics both attributes and methods or operations. Therefore, both 'Baby' and 'Teenager' class will also have the attribute 'name' and 'talk' method. In the context of polymorphism we will focus on the method within the class. In this case, the 'talk' method will produce a different response depending on the class where it is implemented. In the 'Baby' class it will respond with some kind of baby talk for example 'ma ma ma ma' since a baby still has a very limited vocabulary. Meanwhile, in 'Teenager' class it will respond differently since a teenager already has many vocabularies.

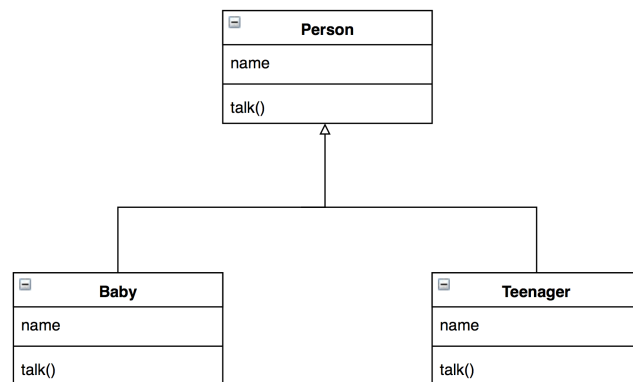


FIGURE 2.8: Polymorphism example

The concept of polymorphism is closely related to the concept of *inheritance* [65]. In the next subsection, we discuss the concept of the inheritance in the context of the object-oriented paradigm.

2.4.2 Inheritance

Basically, *inheritance* defines a relationship between classes where a class share its characteristics or behavior to other classes. Semantically, inheritance denotes an 'is-a' relationship [67], for example, a cat *is a* type of an animal where an animal *is a* type of a living creature.

By definition, the term inheritance can be defined as a mechanism which allows new classes to be defined based on existing classes; a new class can be defined as a specialisation of one that has already been defined. In this case, the specialised class inherits the characteristics (attributes and methods) of the class it is created from. This specialised-class is also known as a sub-class or child-class, while the existing class that inherits its characteristics is also known as a general-class, super-class, or parent-class [68].

There are two types of inheritance: single and multiple [67–69]. Single inheritance can be described as a condition where a child-class has exactly one parent-class. For example, in Figure 2.9 (a), a child-class "B" has only one parent-class "A". Meanwhile, multiple inheritance can be described as a condition where a child-class has more than one parent-class.

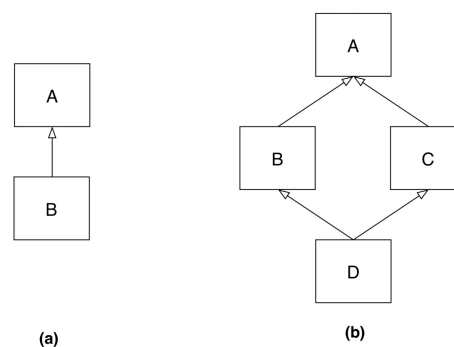


FIGURE 2.9: (a) Single Inheritance; (b) Multiple Inheritance

2.4.3 Visual inheritance

The concept of inheritance in object-oriented paradigm is commonly adopted in graphical user interface design. This concept is being adopted in order to maximise efficiency in the design process, such as in the case when we have a common (base) design that can be reused further by inheriting this base design [70, 71].

For example, in creating a visual representation of a generic dialogue window in the context of UI Frame design. We may create the visual representation of a generic dialogue window, for example, as a rectangle that is equipped with a button with a letter 'X' for a basic function such as 'Close' for closing the dialogue window (as illustrated in Figure 2.10). This visual representation can be reused by more specialised frame, e.g., Login frame, by implementing visual inheritance mechanism. As illustrated in Figure 2.10, we may reuse the visual representation of the generic dialogue window for creating the Login dialogue window. We may, for example, add input fields for the 'Username' and the 'Password' as well as 'Reset' and 'Submit' button to visually represent the Login dialogue window.

The concept of inheritance has inspired the idea of visual inheritance design in defining the design principles. The concept of inheritance offers efficiency in the design process and it also facilitates

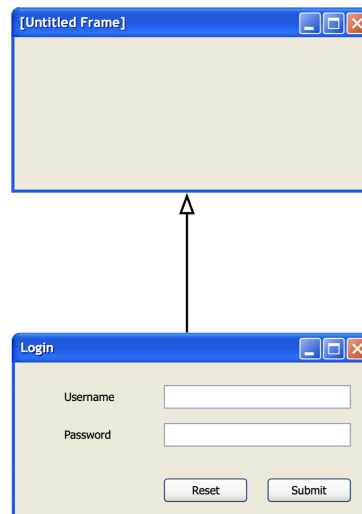


FIGURE 2.10: Example of the visual inheritance in User Interface design

design coherence between related classes, for example, as illustrated in Figure 2.10. The main idea of visual inheritance therefore can also be applied in the process of visual notation design based on metamodel since in a metamodel we can identify the relationship between classes including their characteristics.

2.5 User-centered design

In this section, user-centered design approach is explained. We start by describing the basic concepts and then continue explaining the activities along with the method that can be used in each activities.

2.5.1 User-centered design concepts

By definition, User-Centered Design (UCD) is a design approach that is based upon an explicit understanding of users, tasks, and environments [72]. UCD can be used for designing a product based on the user's perspective. Therefore, this approach offers a potential solution for us to conduct this research in terms of developing a notation that considers the user's feedback as one of the inputs.

UCD suggests that users be actively involved in both design and evaluation process [73]. Besides that, the users who are involved should be relevant, have capabilities and experience related to the product that is being designed. The effectiveness of user involvement increases as the interaction between the developers and the users increases [72].

2.5.2 User-centred design approach

UCD emphasises an iterative process. It should be used to eliminate uncertainty during the development of a product progressively. Iteration means that the product specification, description, and

prototypes are revised and refined when new information is obtained in order to reduce the risk of the product under development failing to meet users' requirements [74].

ISO 9241-210 provides generic guideline for performing UCD process that could be adopted in this research since the design process emphasise in understanding users' needs and context of use. According to this guideline, before conducting the UCD activities, the stakeholders of the product-to-be should be identified and then the following activities can be iteratively performed [74]:

- 1. Understand and specify the context of use.**

The characteristics of the users such as roles, tasks, and experience define the context in which the product is used. It is useful to gather and analyse information on the current context in order to understand, and then specify, the context that will apply to the future product.

Analysis of existing condition of the product provide information such as deficiencies and also can reveal needs, problems and constraints. There are several methods that can be used in this activity: stakeholder meeting, survey on existing users, field study/observation.

- 2. Specify the user requirements.**

The requirements derived from user needs and the context of use, therefore users and other stakeholder needs should be identified. The stakeholder meeting or stakeholder survey methods as mentioned above can be used as an example to conduct this activity.

- 3. Produce design solution.**

Potential design solution are produced based on users' requirements (needs and context of use). Beside that further user requirements that can emerge as potential design solutions are detailed and evaluated. To conduct this activity, user participatory design method can be implemented by involving the users in design process. This method is primarily used in UCD since it is align with the core concept of UCD where the users need to be involved in the design process [75].

In visual notation research, this participatory design method has been implemented as noted in [51] where the author asked the participants of their research to produce the design for the visual notations. In relation to this research, this method was adopted in this research where the stakeholders were invited participate in the development of the proposed SACM argumentation notation.

- 4. Evaluate the designs against requirements.**

User-centred evaluation (evaluation based on users' perspective) is a required activity in UCD. It must be conducted both to obtain early feedback in order to improve the product and, at a later stage of process, to determine whether the requirements have been satisfied.

UCD provides user-based evaluation method where the users need to be involved in order to provide feedback about the design product. The users' feedback later on will be the input for conducting refinement activity where the product need to be improved in order to satisfy the users' need [74].

The above UCD activities generally similar with activities that are noted in others UCD literature such as [72] and [73] where they also emphasise in iterative process and product refinement in order to satisfy the users' needs. Beside that, in designing a visual notation, UCD approach has been adopted in previous studies. For example, as discussed in [51] where the authors adopt UCD approach in re-designing UML user-based notations. Based on the result from their experiment, the UML notations that design by users' provides better comprehensibility aspect rather than existing notations.

Based on the above explanation, we may conclude that UCD offers potential solution for conducting systematic elicitation process for this research since UCD methods such as stakeholders' survey is very useful for us to understand users' opinion.

2.6 Summary

In this chapter, we have explained the research context with several relevant literature from the different field of knowledge areas such as visual notation research, structured assurance cases, visual inheritance concept in the object-oriented paradigm and user-centred design.

We have reviewed the basic concepts of visual notation and identified current best practice in developing and evaluating a visual notation along with the existing approaches and methods. Based on our review, currently only a few works in the literature focus on providing a guideline in designing a visual notation for a predefined metamodel. As the current metamodeling approach is widely adopted as a mechanism for improving standardisation and interoperability in modelling languages [17], it is important to provide specific guidelines that focus on the design of visual notation for a predefined metamodel.

In the structured assurance case research area, we have reviewed the structured assurance case concepts and its representations. The methods for structuring assurance cases such as GSN, CAE, and SACM argumentation were explored. We found that both GSN and CAE adopts graphical approach in representing the assurance case. However, the concept of SACM argumentation that is specified in SACM argumentation metamodel, currently, is not equipped with any visual notations. It would be beneficial if we provide the visual notation for the SACM argumentation metamodel. With this aim, we need a systematical approach in developing the notations, however, currently only a few works in the literature that provides a specific guideline in designing a visual notation based on a metamodel. Therefore, we explored the concept of user-centred design and the visual inheritance in object-oriented paradigm.

The concept of visual inheritance inspired us in developing a systematical approach for designing a set of visual notation based on a metamodel. This concept emphasises the design coherence between parent and child-classes, where the basic design characteristics of a parent-class must be inherited to its child-classes. On the other hand, the concept of user-centred design provides as knowledge in involving user in the design and evaluation activities when developing a product in this case is a visual notation.

Chapter 3

Visual notation design approach

3.1 Introduction

In this chapter, the proposed visual notation design approach is described. This approach is developed in order to be used to design the SACM notation since there is a lack of visual notation design approach that can be adopted, specifically in the context of designing a visual notation based on a defined metamodel.

In Chapter 2, we have discussed several existing notation design approaches. Due to the adoption of a metamodeling approach as part of the advancement of a modelling language (e.g. to support interoperability), in this thesis, we specifically propose an approach for designing a visual notation based on a defined metamodel of a modelling language. Therefore, in this approach, the primary input is the metamodel of the language. The metamodel in this context must have the description of the semantics of the elements and the hierarchical structure (in graphical form, e.g. using UML approach) of the elements. Both the semantics and the hierarchical structure of the elements will be used to create the visual representation of the element in the visual notation design process. Therefore, the metamodel without the semantics and the hierarchical structure of the elements would not be suitable as the input for our approach.

It is important to note that the validity of the metamodel, including its specification is not part of the scope of the proposed approach in this study. The focus of the proposed approach is to develop a visual notation based on a defined metamodel that provides a representation for the abstract syntax of a modelling language.

In the next sections, the overview of the proposed approach is described along with the steps to develop a notation.

3.2 Overview

The objective of the proposed visual notation design approach in this thesis is to produce an *effective* notation. In this thesis, we define the effectiveness of a visual notation using the following characteristics, which are inspired from the visual notation design literature [3, 48, 53, 54]:

- **Accurate:** a notation that can be used by its user to construct a model (diagram) that is correct in every detail. This characteristic is essential for effective communication through a visual medium (i.e. diagram) in order to deliver a valid message from the creator of the diagram to the reader.
- **Easy to learn:** the visual representation of the elements should be easy to identify by its users so that the users can easily read the diagram that consists of multiple elements of the notation.
- **Easy to use:** the users can use (draw) the visual representation of the elements quickly when they need to create a diagram using the notation. This characteristic is derived from the definition of notation efficiency (from [3]) to measure the user's speed when using the notation to create a diagram. Therefore, we consider this characteristic to be important for a notation so that its users can easily create (draw) a diagram using the notation.

In order to evaluate this objective, the notation produced using the proposed approach needs to be assessed. The assessment can be done by comparing the effectiveness of the notation that is designed using the proposed approach and another notation that is designed using another approach.

In order to ensure the notation has these characteristics, the following design approach was developed. There are two preparation steps before starting the design process that needs to be considered when applying the proposed approach:

- Consider the design scope of the metamodel
- Consider the involvement of the users of the notation

The first preparation step is related to the identification of the design scope of a metamodel. This first step may need to be conducted in the case of designing a notation for a complex metamodel (i.e. a metamodel that is composed of several sub-concepts) such as UML. The UML metamodel consists of multiple sub-concepts like Behaviour and Structure diagrams, where each diagram has its specific types of diagram such as Class diagram (part of Structure diagram type) and Activity diagram (part of Behaviour diagram type). When designing a visual representation of a complex metamodel, it may be necessary to prioritise which part of the metamodel to design the notation for first. Prioritising the design of the part of a complex metamodel may help the notation designer to focus on one task at a time in the design process.

The second preparation step is related to the involvement of the notation user in the design or evaluation phase. Developing a visual notation involves creating a product that is definitely user-centric, so users of the product should be involved where possible in the development and evaluation phases. An example of an experimental study that reported the involvement of the notation users both in the design and evaluation phases can be found in [51]. That research aimed to investigate the semantic transparency of the UML notation by conducting an experiment involving the notation user in the design and evaluation phase of the notation. In the design phase, a group of notation users was asked to create an alternative notation for the UML concepts. In the evaluation phase, another group of notation users were asked to assess the semantic transparency of the original UML notation and the created alternative notation. Based on this experiment, it was seen that involving the notation users in the design phase can be more complicated than involving the notation users in the evaluation phase.

Figure 3.1 illustrates our view regarding the life cycle of a visual notation. We see the development and evaluation of a visual notation as an iterative process. After the development process is completed, the evaluation phase needs to be conducted. The result of the evaluation phase, e.g. the feedback from the notation users, can be used as an input for the development phase in the next iteration process in order to improve the notation.

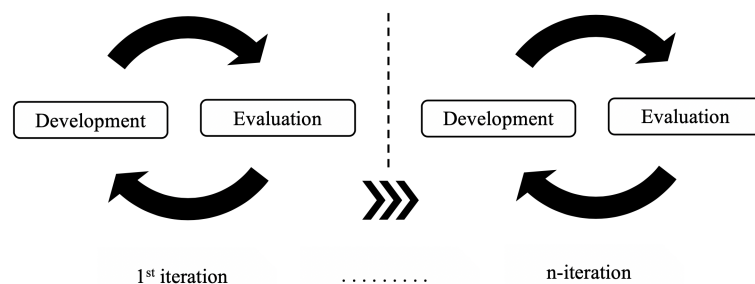


FIGURE 3.1: The visual notation life cycle

In this thesis, following the development process of a visual notation as shown in Figure 3.1, we propose an approach that considers a defined metamodel as part of the process. The approach is shown diagrammatically in Figure 3.2 that can be used to create a visual notation in the development phase of the visual notation design life cycle. The following four steps are involved in the proposed visual notation design approach in the development phase of a visual notation in this thesis, where the metamodel is used as an input for the first task (create a design path):

1. Create a design path (based on a defined metamodel)
2. Identify design constraints
3. (for each identified element in the design path) Create the visual representation
4. Document the design rationale

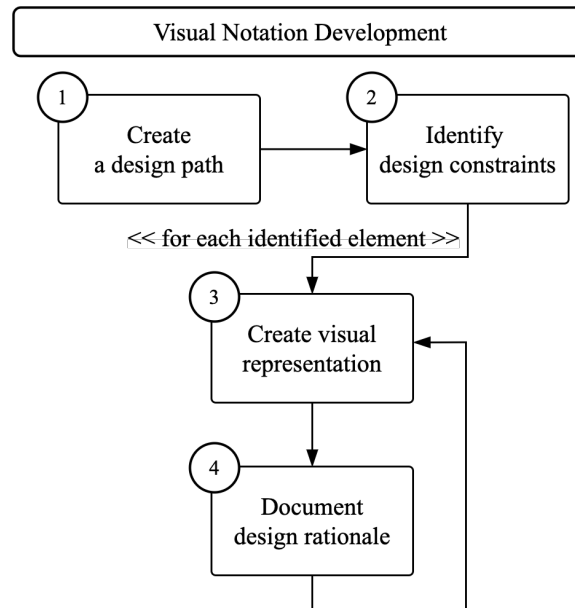


FIGURE 3.2: The proposed visual notation design approach

The following inputs are required to apply the proposed approach:

1. A metamodel with the description of the semantics of the elements and the hierarchical structure of the elements for creating the design path and creating the visual representation of the elements.
2. Existing notations (e.g., in a similar domain to the notation that will be developed) as the input to define the design constraints in creating the visual representations of the elements.
3. Visual notation design theories and principles such as visual variables, semantic transparency, and notation usability to help to define design constraints.

Figure 3.3 illustrates the details of each step along with the expected outputs. The rounded rectangle with shaded colour represents the general step of the approach. The rounded rectangle within the dashed line represents the sub-step as part of the general step. The parallelogram represents the output (Ou) of the steps. The arrowhead line represents the flow of activity related to the approach.

The first step in the proposed visual notation design approach is related to the creation of a design path. A design path provides the order in designing or creating the visual representations of the metamodel elements and also provides information related to the elements that are involved in the design process. Step 1, Create a design path, consists of three sub-steps: (1.1) Metamodel elements identification; (1.2) Elements prioritisation; and (1.3) Design path creation. First, the input for sub-step 1.1 is a defined metamodel. Next, the sub-step (1.1) output is used as the input for sub-step 1.2, and the output of the sub-step (1.2) is used as the input for sub-step 1.3. Finally, the output of sub-step 1.3 is the created design path (Ou.1).

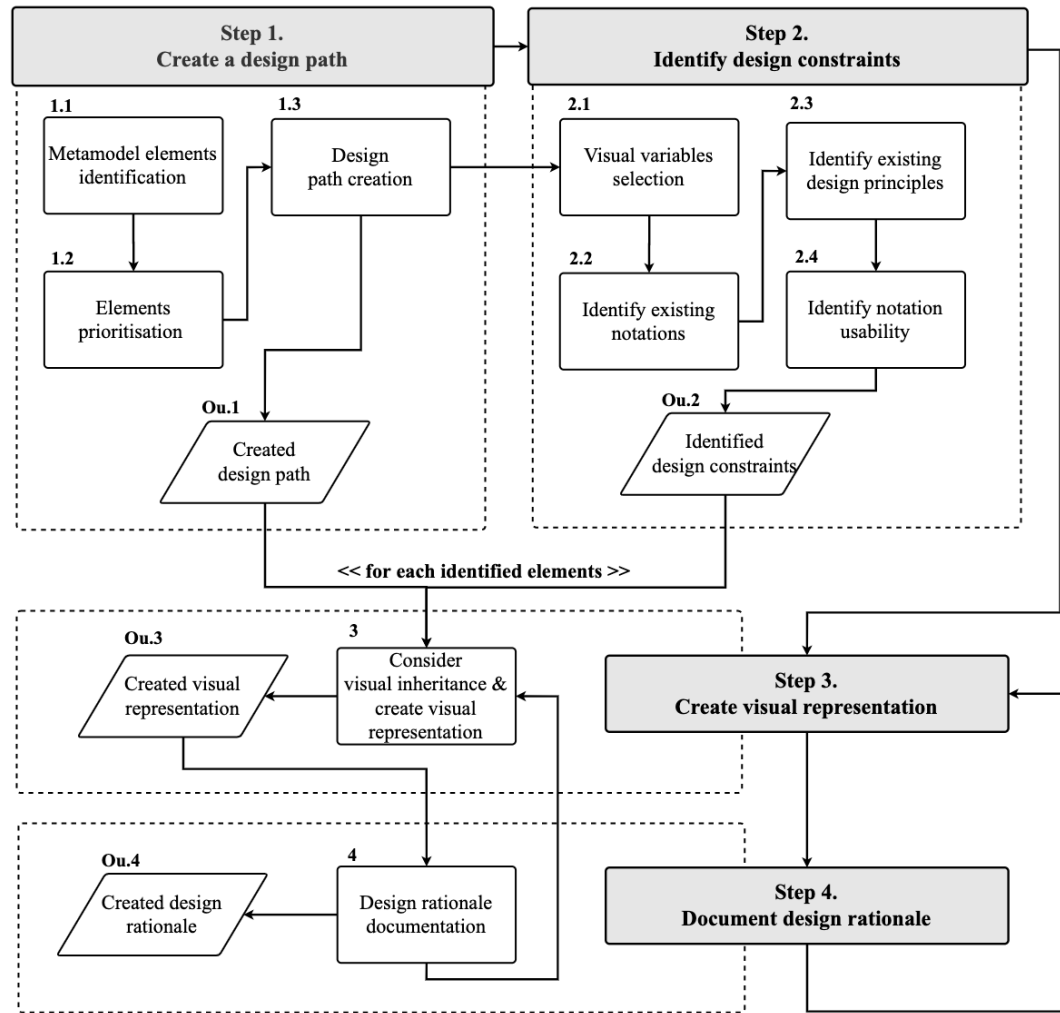


FIGURE 3.3: Visual notation design process

After completing step 1, the process continues to step 2: Identify design constraints. This step 2 consists of four sub-steps: (2.1) Visual variables selection; (2.2) Identify existing notation; (2.3) Identify existing design principles; and (2.4) Identify notation usability. The output for step 2 is the identified design constraints (Ou.2). The identified design constraints are used as input to create the visual representations in step 3.

The output from steps 1 and 2 are used as an input for step 3: Create visual representation. The order in creating the visual representation of the elements is based on the created design path (output of step 1). In creating the visual representation, we need to respect the design constraints that we have considered in step 2. In step 3, to create each visual representation of the identified elements, it is important to consider visual inheritance in order to show family resemblance among related visual representations (of elements that have similar semantics). It is also important to check the visual distance between visual representations in order to avoid creating two or more identical visual representations that can cause visual representation overload. The output of step 3 is the created visual representation of the identified elements (Ou.3).

The design rationale of each created visual representation needs to be documented in Step 4. Thus, the creation of the visual representation and the documentation of the design rationale is a recursive process until all the visual representations of all the identified metamodel elements are created.

Further explanation regarding each step shown Figure 3.3 and the application example are described in the following sections.

3.3 Step 1: Create a design path

The first step in the proposed visual notation design approach is to create a design path. A design path is the order in designing or creating the visual representations of the metamodel elements. It helps to indicate the elements that are to be visually represented using the design process. Since a metamodel consists of a number of elements, we also need to decide which elements need to be designed first and which element is next in order since another element's visual representations can influence the creation of another visual representation through the visual inheritance mechanism.

Figure 3.4 illustrates the input, process, and output related to the creation of a design path.

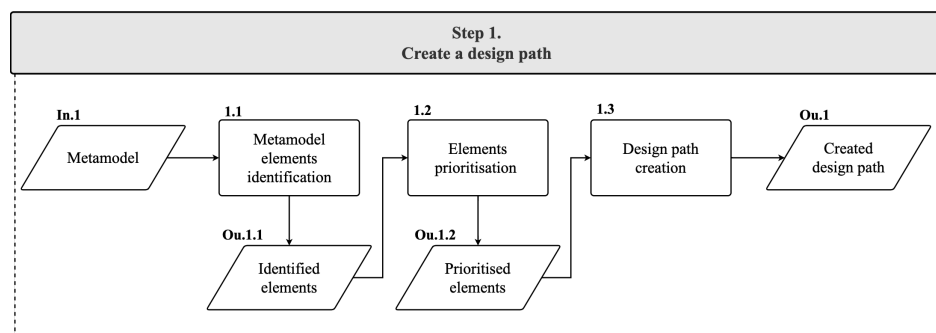


FIGURE 3.4: Step 1: Create a design path

The input to create the design path is a given metamodel (In1). The elements of the metamodel that are considered necessary to have a visual representation will be identified in the first sub-step (1.1). After the elements have been identified (Ou.1.1), we need to prioritise these elements (1.2) in order to decide which element needs to be designed first. This sub-step is related to the utilisation of the visual variables in the creation of the visual representation. The output of this sub-step is a list of the prioritised elements (Ou.1.2). After the list of the prioritised elements is created, we need to create a design path to show elements of the metamodel that are to be included in the creation of the notation, for example, from the root class to the all the identified elements. After completing this sub-step we will have the created design path (Ou.1).

The details of each substep is described in the following subsections along with its related example.

3.3.1 Metamodel elements identification

In order to create a design path, the first thing to do is to identify which metamodel elements will be represented using a visual representation. We need a defined metamodel as an input for completing this sub-step.

A metamodel that is specified using UML, for example, typically consists of several elements such as Classes, Attributes, and Relationships. A Class can be either an abstract or a concrete one. An abstract class is a class that cannot be instantiated at the model level. A concrete class is a class that can be instantiated at the model level. Therefore, a concrete class is a class that can be represented using a visual representation. A class in a metamodel sometimes has attributes that can be inherited to its sub-classes. These attributes also can be represented using a visual representation. Classes can be connected using a specific type of relationship. A relationship can be either an Association, Aggregation, Composition, or Generalisation. An Association relationship can be represented using a visual representation, while Aggregation and Composition relationships can be used to define compositional rules of the element in a diagram. The Generalisation relationship, in our approach, is used to define the visual inheritance relationship between elements.

In some cases, a metamodel can be composed of a huge number of elements (e.g. UML metamodel). In such cases, we can iteratively develop the notation, or in other words, we need to focus on developing a visual notation for one particular part of the metamodel at a time, and iteratively developing the notation for the other parts of the metamodel. In order to do so, we need to identify which metamodel elements are considered necessary to be designed first by comparing the semantic importance of the metamodel elements related to the application domain. In order to complete this activity, the notation designer needs to get a better understanding of the metamodel itself by studying the the purpose of the metamodel, context of use, semantics and relationships of each metamodel element. Studying the metamodel may provide the knowledge related to which elements are used to represent the core concepts of the language specified within the metamodel. It is also important to study existing notations in the domain related to the metamodel. Studying existing notations may provide insight in determining which elements are necessary to have in a visual representation. However, it is important to note that different languages in the same domain can have different concepts that are specified in their specific metamodel.

Let us consider Figure 3.5 to illustrate the metamodel elements identification process.

Figure 3.5 illustrates an example of a Metamodel (referred to as Metamodel X) that is graphically represented using a UML Class Diagram approach to specify the concept of modelling language M. In this metamodel, Class A is the root class that is defined as an abstract class (indicated by a *). Class A has three sub-classes which are Class B, C, and D. Class B is defined as an abstract class that has two sub-classes namely Class E and F. Class F is defined as an abstract class that has three sub-classes which are Class G, H, and I. Meanwhile, Class C has no sub-classes, and Class D has two sub-classes

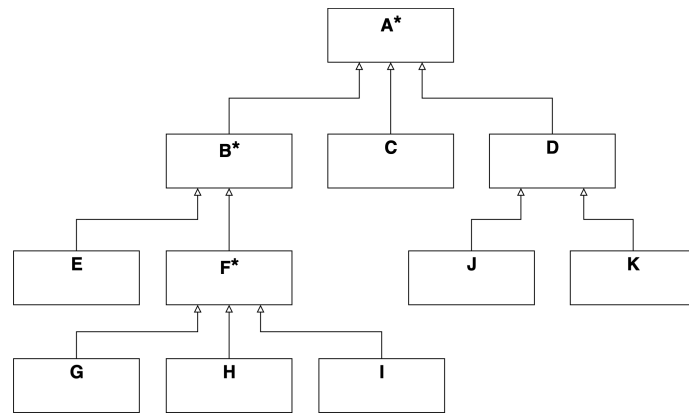


FIGURE 3.5: Metamodel X

namely Class J and K. Other than Class A, B, and F, the other classes are defined as a concrete class. Based on this information, we identified that we can create a visual representation of Class E, G, H, I, C, D, J, and K since they are concrete classes.

Supposed after we study the Metamodel X, we get the following information:

- After studying the Metamodel X and its background modelling language concept, it is possible to create a model by utilising only the metamodel elements that are representing the core concept of the modelling language M, in this case the Classes A, B, C, E, F, G, H, and I. The other classes such as Class D, J, and K are not mandatory elements for creating a model based on modelling language M since these elements are defined as supporting elements.
- After studying another existing modelling language (e.g., modelling language N) in the same domain as modelling language M, we find that the modelling language N only provides its user with the visual representations of the elements that are used to represent the core concepts of the language. Other literature also stated that the supporting concepts are rarely implemented or used in the model.

Based on the above considerations, we may consider focusing on creating the visual representations for the core concepts of the modelling language M that is specified in Metamodel X by adopting an iterative visual notation design approach. The visual representation of Classes C, E, G, H, and I, that represents the core concept of the language, will be designed in the first iteration of the notation development. The rest of the elements, Class D, J, and K that represents the supporting concept of the language, will be developed in future iterations of the notation development process.

In another case, if we decided to ignore the consideration points as previously described and decided to create the visual representations of the elements that represents both core and supporting concepts of the language, we may create the visual representations for all concrete classes in the metamodel (i.e. create a visual representation of Class C, E, G, H, I, D, J, and K). The Generalisation relationships

(used to connect between Classes in the Metamodel X), are used in our approach to indicate visual inheritance that will be used in the creation of the visual representation of the elements.

To summarise, the first thing to do to create a design path is to identify the metamodel elements that need to be represented using a visual representation. Concrete Class, Attributes, and Association relationship are metamodel elements that can be represented using a visual representation. The Aggregation and Composition relationships can be used to define compositional rules of the visual representations in a diagram. The Generalisation relationship, in our approach, is used to define the visual inheritance relationship between elements. For a metamodel that is composed of a large number of elements, we may adopt an iterative visual notation design approach by identifying metamodel elements that are considered necessary to be designed first. The visual representation of the rest of the elements can be designed in future iterations. The expected output of this sub-step is the list of the identified elements that will be represented using a visual representation.

3.3.2 Elements prioritisation

After we have a list of the identified metamodel elements, we need to undertake an element prioritisation process. Prioritising the metamodel elements helps in considering the utilisation of the visual variables. Shape, Position, and Colour are examples of visual variables. Each visual variable has a unique value. For example, the value of the Shape visual variable can be Rectangle or Circle, and for Colour visual variable can be, for example, Red or White. These visual variables and their value are the main components in developing a visual representation of an element.

A visual representation of an element must be unique in terms of the utilisation and combination of the values of the visual variables. Deciding to use a specific combination of visual variable values to represents an element means that we cannot use the same combination to represents other elements in order to avoid visual representation overload. Visual representation overload is an anomaly that occurs when more than one element (or construct) is represented by an identical visual representation. Visual representation overload leads to ambiguity and the potential for misinterpretation for the users of the notation (further description and example about visual representation anomalies are described in Section 2).

Creating a list of a prioritised metamodel elements means that we will create a visual representation of an element by combining visual variable values, and this combination cannot be used by other visual representations for other elements on the list.

In our approach, we suggest that the importance of the semantics (related to the application domain) of an element relative to the other elements in the metamodel as the main consideration in the prioritisation process of the elements. Studying the metamodel, its specification, and its domain is the best way to identify the role of an element in a metamodel.

Let us consider the output from the example described in sub-step 1 as the input of the element prioritisation process.

Supposed we have a list of the identified elements that need to be represented using a visual representation from sub-step 1 as follow:

- Class C
- Class E
- Class G
- Class H
- Class I

We need to prioritise these elements by comparing the semantic importance that is related to the application domain of an element and other elements. Suppose that after the comparison process, we created the following prioritised list:

- Class E
- Class G
- Class H
- Class I
- Class C

The above list indicates that we need to create the visual representation of the Class E first, and then continue to Class G, Class H, Class I, and finally Class C. Having this list (i.e. prioritised elements) means that we also prioritise use of a specific combination of visual variables values to create a visual representation of an element, and that combination cannot be used to visualise other elements.

To summarise, sub-step 1.2. focus on prioritising metamodel elements by comparing the semantic importance that is related to the application domain of an element and other elements. The element prioritisation is important in order to create a unique visual representation for each identified element in terms of utilising visual variables values. The expected output from the element prioritisation process is a list of prioritised metamodel elements.

3.3.3 Design path creation

After we have created a list of prioritised elements, we can create a design path. A design path is the order in designing or creating the visual representations of the metamodel elements. It helps to

indicate the elements that are involved in the design process of a visual notation. Since a metamodel consists of a number of elements, we need to decide which element needs to be designed first and which element is next in order to focus on creating a visual representation of one element at a time. It is also important to know which elements are involved in the design process since an element can influence other elements' visual representation via visual inheritance.

The starting point to create a design path is from the root class of the metamodel. The root class typically is an abstract class that sometimes has an attribute that can be inherited to its subclass. From the root class, we may identify attributes that can be inherited to its sub-classes. From the root class we need to decide which element to consider next. The next element must be the first element from the list of the prioritised metamodel elements or elements that are connected via generalisation relationship to the first element from the list of the prioritised metamodel elements. After that, we can continue to the other elements based on the list of the prioritised metamodel elements.

Let us consider Metamodel X again (as shown in Figure 3.5) to illustrate an example of the creation of a design path. Suppose we decided to focus on creating visual representations for the core concept of the modelling language M. In this case, we want to create a visual representation of concrete Class C, E, G, H, and I (the output of sub-step 1.1). After the prioritisation process, we decided to create the visual representation of the Class E first, and then continue to Class G, Class H, Class I, and finally Class C (the output of sub-step 1.2). This information is used to create a design path. The design path starts from the root Class of the metamodel (i.e. Class A). From the root Class we continue to Class B since it is connected via generalisation relationship to Class E (Class E is the first class on the list of the prioritised metamodel elements that the visual representation of this class is need to be created first). From Class E we continue to Class G (via Class F since this Class is connected to Class G via generalisation relationship), and then to Class H, I, and C as defined on the list of the prioritised metamodel elements.

We illustrate the created design path in Figure 3.6. The Rectangles in Figure 3.6 represent the Classes of the metamodel, the Lines with a hollow arrowhead connecting the classes represent the Generalisation relationships, and the dashed lines with an arrowhead represent the design path. Class A, B, and F, are an abstract class (a star added next to the class name to indicates that this class is an abstract class) that are included in the design path since they can influence the creation of visual representation of other classes, for example providing design constraints that can be inherited to its sub-classes. Class E, G, H, I, and C, are the classes that requires a visual representation that need to be designed (drawn as light-dark brightness). Class D, J, and K, are the classes that are not included in the current iteration of the visual notation design process. If there was another abstract Class between B and F, for example we add another class named as class O, then this class need to be added to the design path because this class is associated directly to the other classes in the design path (Class B and F) that can also provides design constraints that can be inherited to the sub-classes for the purpose of the visual representation creation.

It is important to notice that the order in creating the visual representation must refer to the list of the prioritised metamodel elements. This means the first class on that list must be designed first and continue to the other classes. The design path helps to indicate the elements that are involved in the design process.

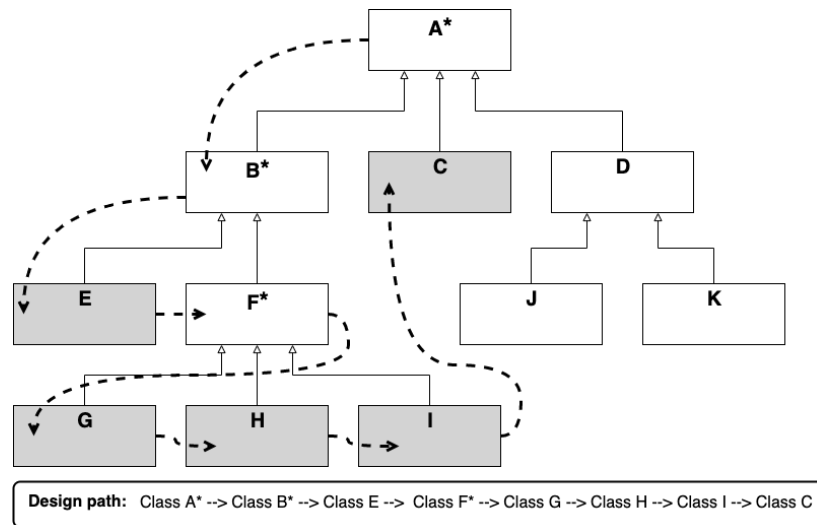


FIGURE 3.6: Example of a design path of the Metamodel X

The expected output of sub-step 1.3 is the created design path. After the design path is created, we need to consider several design constraints in order to create the visual representations of the element. The detailed explanation regarding the design constraints is described in the next section.

3.4 Step 2: Identify design constraints

Design constraints are the scope and limitation statements that are defined by the notation designer in designing a notation. A design constraint also can be seen as a requirement that must be satisfied in designing a notation [3, 19]. Once a design constraint is planned to be adopted, it needs to be respected in the creation of visual representations.

As illustrated in Figure 3.7, in our approach, several design constraints can be considered in designing a notation: the selection of visual variables (2.1), consider existing notations (2.2), consider existing visual notation design principles (2.3), and notation usability (2.4). These design constraints are adopted from the existing literature related to the theory of visual variables (In.2.1), existing notations in the related domain (In.2.2), visual notation design theories and design principles (In.2.3), notation usability and user-centred design (In.2.4). The information from the literature is used as the input in this step. The details of each design constraints are described in the following sub-sections. As the expected output from this step is a list of considered design constraints (Ou.2) that will be used in the creation of the visual representations.

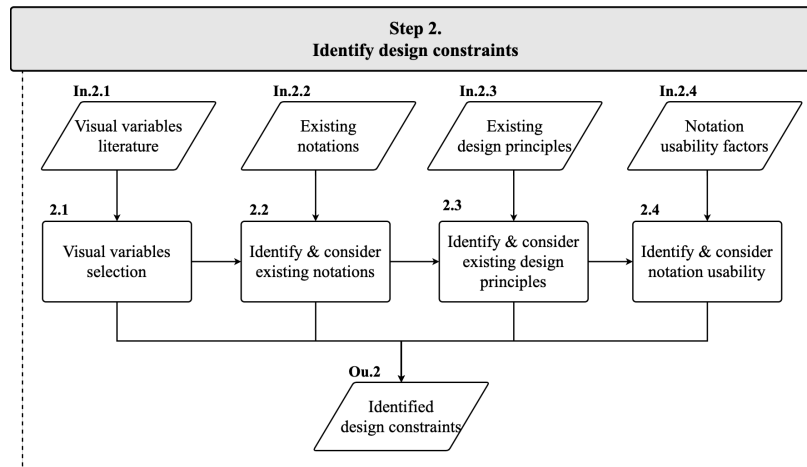


FIGURE 3.7: Step 2: Identify design constraints

3.4.1 Visual variables selection

The visual variables are the basic building block of graphic visualisation that can be used to graphically encode information in a visual representation [3, 4, 76]. A visual representation can be created by utilising and combining visual variables and their values such as Circle, Rectangle, or Triangle as the unique value of Shape, and Red, Blue, or Green as the value of Colour.

To create the visual representations of the metamodel elements, we need to select which visual variables that will be used by considering the importance and limitation of the visual variables, e.g., the importance of Shape, or limitation of Colour. The selected visual variables must be respected when designing the visual representations and can be inherited when designing visual representations with similar semantics through visual inheritance process.

The visual notation designer can develop many different visual representations of notation by selecting and combining the following visual variables in different ways:

- **Position/Location (vertical and horizontal):** describes the position of the visual representation relative to a coordinate frame such as changes in the x,y location (in 2d). The Position is useful in visual notation design, for example, to indicate the position of a visual representation relative to another visual representation.

Let's consider Figure 3.8 as an example of the role of visual variable Position to describe the position/location of a visual representation relative to another visual representation.

Figure 3.8 illustrates a Use Case diagram with a subject (sometimes called a system boundary) is Books Online with applicable use cases (Browse, View, and Purchase Items) and a Customer actor. The position of the Customer in this example is located horizontally relative to the rectangle that is representing the Subject, and the position of the use cases is located within the Subject.

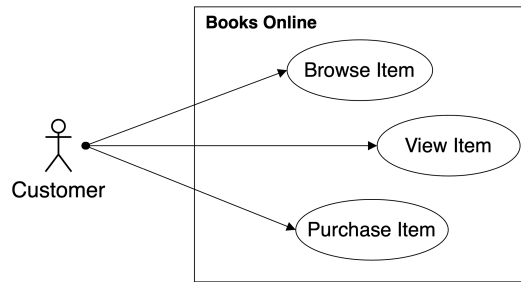


FIGURE 3.8: Example of Location/Position in Use Case diagram

- **Size:** describes the amount of space occupied by the visual representation such as changes in length, area, or repetition. For example, the size of the rectangle as shown in Figure 3.8 is 390 pt (width) and 317 pt (height).

Size can also be used to provides emphasis to a particular visual representation, e.g., to signify importance - such as the thickness of a line or outline of shape (since it occupies some space or area). For example, the size of a line that represents a Join in the UML State Machine notation as illustrated in Figure 3.9.

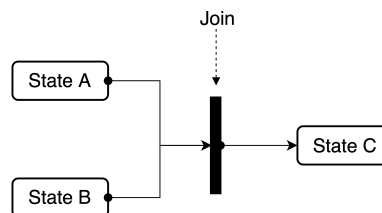


FIGURE 3.9: Example of Size indicated by the thickness of the line used for Join

In Figure 3.9, the size of the (thickness of the) line is 12 pt, and the size of the arrowhead line is 1 pt.

- **Shape:** describes the external form (i.e., the outline) of the visual representation. Shape can vary from highly abstract, such as circles, squares, or triangles, to highly iconic, directly mimicking the referent represented by the visual representation.

In the context of visual notation design, of all visual variables, Shape is the most important visual variable for discriminating between visual representation, as it represents the primary basis on which humans identify objects in the real world. Moreover, it has the largest range of values amongst the visual variables [3, 24, 59, 76, 77]. Not limited to software and system engineering field, many notations also utilise visual variable Shape from different families (e.g. Rectangle, Circle, and Line) to denote or visually represent their constructs such as used in UML, BPMN, and DeMarco Data Flow Diagrams (DFDs).

In Figure 3.10, we can see that in DeMarco DFDs, the visual variable Shape is used with different values to represent the different constructs of DeMarco DFDs. For example, Shape with

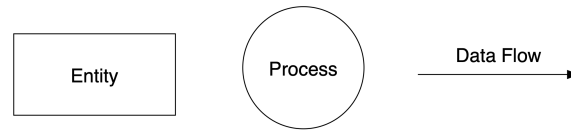


FIGURE 3.10: Example of DeMarco Data Flow Diagram visual representations

value 'Rectangle' is used to represent 'Entity', Shape with value 'Circle' is used to represent 'Process', and Shape with value 'Line' and 'Triangle' combine together to represent 'Data Flow'. Other than the usage of visual variable Shape, another visual variable is also used in this example of DeMarco DFDs, such as the usage of visual variable Brightness with value 'Light' as the background within the Rectangle and Circle, and also visual variable Brightness with value 'Dark' as the background within the Triangle. Utilising and combining different types of visual variables and values can be useful in order to create different types of visual representation.

- **Orientation:** describes the direction or rotation of the visual representation from 'normal'. The normal orientation typically is relative to the designer's defined baseline.

Let us consider Figure 3.11 as an example of Orientation in visual representation.

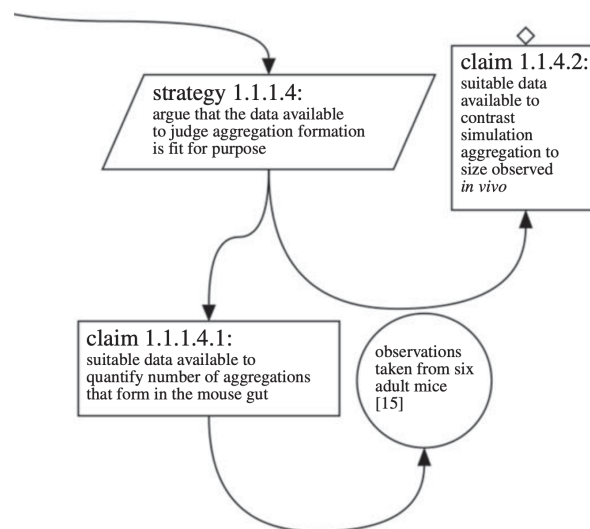


FIGURE 3.11: Example of Orientation in visual representation: a fragment of an argument structure adopted from [5]

Figure 3.11 illustrates a fragment of an argument structure using GSN-like notation as presented in [5]. The Claim 1.1.4.2 is drawn rotated 180 degrees from its normal representation. This Claim is an Undeveloped Goal in GSN and supposed to be drawn with a rectangle with the hollow-diamond as an undeveloped element decorator placed at the centre-bottom, presents a claim which is intentionally left undeveloped in the argument. However, in this example, the visual representation of the undeveloped Goal is drawn rotated 180 degrees, which in this case,

violates the rule as defined in the GSN community standard in drawing the undeveloped Goal and may confuse GSN users.

- **Colour (hue):** describes the dominant wavelength of the visual representation on the visible portion of the electromagnetic spectrum such as blue, green, red. Colour changes in hue at a given value [78].

In notation design, Colour is commonly used to implement dual coding, i.e., using multiple visual variables for designing a visual representation in order to increase perceptual discriminability between visual representations (distinguishing between visual representations) [24].

The usage of Colour in designing a notation can be useful, such as, may increase perceptual discriminability between visual representations. However, the notation designer also needs to be careful when considering the use of Colour in notation design since Colour can be sensitive for some context of use. For example, Colour could cause different visual perception for some users with a limitation such as colour blindness; and also could cause different result in visual perception such as when printing diagrams that used colour when printed on black-and-white printers [3].

Further explanation regarding the benefit and limitation of Colour in notation design is described in the Notation Usability design constraint sub-section along with some examples.

- **Brightness (colour value):** describes the relative amount of energy emitted or reflected by the visual representation. Variation in color value results in the perception of shading or areas of relative light and dark, therefore, color value is also referred to as ‘lightness’ in color theory [78].

Let us consider Figure 3.12 to illustrates an example of Brightness in notation design.

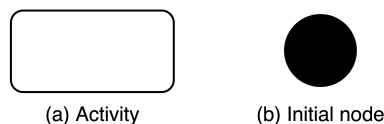


FIGURE 3.12: Example of Brightness in visual representation

As shown in Figure 3.12, the Brightness of an Activity visual representation in a UML Activity diagram is defined as Light, the Brightness of the Initial node visual representation is defined as Dark.

- **Texture:** describes the coarseness of the fill pattern within the visual representation. For example, the visual representation of the Message in UML Sequence diagram that is represented using an arrowhead with solid (texture) line, and the visual representation of Return message that is represented using an arrowhead with a dashed line (as shown in Figure 3.13).

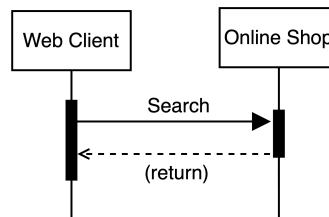


FIGURE 3.13: Example of Texture as used in UML Sequence diagram

3.4.2 Identify existing notations

In creating a notation, it is important to identify and consider other existing notations in order to avoid the occurrence of a notation clash that can confuse the notation user. The clash between notation in this context is related to the usage of an identical visual representation to represent two different elements with different semantics. If such cases occur, they could confuse the user of the notation, even more, so if the notation is in the same application domain.

Let us consider Figure 3.14 as an example of the usage of the same visual representation to represent two different elements with different semantics from two different notations.

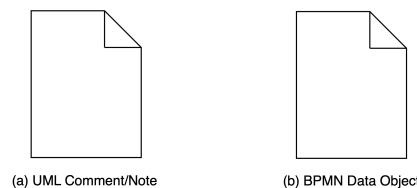


FIGURE 3.14: Example of the same visual representation used to represent two different elements with different semantics

Figure 3.14 illustrates the UML Comment or Note (a) and the BPMN Data Object (b) visual representation. Their visual representation is identical, however, they have different semantics. The UML Note is a construct that can be used to add further textual information for the user such as a comment or constraint. The BPMN Data Object provide information about what activities require to be performed and/or what they produce - Data Objects can represent a singular object or a collection of objects.

Consider another example of notation clash as illustrated in Figure 3.15.

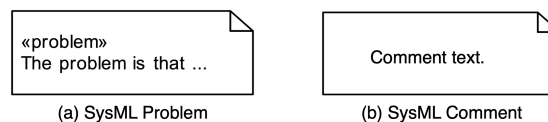


FIGURE 3.15: Example of the same visual representation used to represent two different elements with different semantics

Figure 3.15 (a) illustrates the visual representation of the SysML Problem and the Figure 3.15 (b) illustrates the visual representation of the SysML Comment. Both of these visual representations

are identical, they can be differentiated only by the text ‘«problem»’ that is placed within the visual representation of the SysML Problem. Text cannot be used as the only aspect to differentiate visual representations because text is not a visual variable. Visual representations that differ only on textual characteristics are technically homographs which means they are identical and must not be used to visually represent the semantics of two (or more) different elements [3].

Based on these examples, existing notations should be considered by the notation designer in notation design to avoid notation clash and confusion for the notation users.

3.4.3 Identify existing design principles

There are several visual notation design principles that can be considered by the notation designer in notation design. The following design principles are adopted from existing visual notation literature such as the Physics of Notation [3], Peirce’s theory of sign [79], and visual notation design studies such as [15, 18, 48]:

1. Semiotic clarity of the notation

Semiotic clarity is a requirement that should be considered when designing a visual notation. It suggests that there should be a one to one correspondence between the semantic constructs of the language and the visual representations [3, 15]. In order to satisfy the semiotic clarity, in designing a visual notation, visual notation designer is suggested to avoid the following anomalies:

- (a) Visual representation **redundancy**, occurs when multiple visual representations can be used to represent the same semantic construct.
- (b) Visual representation **overload**, occurs when two different constructs (or more) can be represented by the same (identical) visual representation.
- (c) Visual representation **excess**, occurs when there are visual representations that do not correspond to any semantic construct.
- (d) Visual representation **deficit**, occurs when there are semantic constructs that are not represented by any visual representations.

2. Semantic transparency

Semantic transparency is the creation of a visual representation whose appearance suggests or represents its semantic meaning [3, 15, 19]. In some cases, not all visual representations can be used exactly represents the semantic of the metamodel element, or even there may be a visual representation that can be used to represent the semantics, but the visual representation can be hard to draw for some users.

For example, the visual representation of the Defensive State of the extended UML Statecharts notation to model security aspects as discussed in [6]. A Defensive State can be used to represent a state of a system when it is expecting or experiencing an attack. The system executes the necessary defensive mechanisms to fend off attacks. Figure 3.16 illustrates the visual representation of a Defensive State.

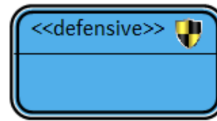


FIGURE 3.16: Defensive State visual representation [6]

For some notation users who create/sketch a diagram or model using conventional approach such as using pen, paper, or a whiteboard, the visual representation of the Defensive State may cause difficulty, i.e., drawing the visual representation, since it uses different Colours and a Shield icon.

To help the notation designer create a visual representation of an element and comply with the semantic transparency principle, we adopt the Peirce theory of signs by categorising the visual representation types to create an intuitive visual representation. Peirce stated that the form of a sign (visual representation) could be classified as follow [79]:

- An *Icon* has similar characteristics to the object that is being referred, e.g., a photograph as it certainly resembles whatever it depicts.
- An *Index* shows factual connection or clue to its object, e.g., using an image of smoke to indicate a fire.
- A *Symbol* provides interpretive habit or norm of reference to its object. Therefore, it needs to be learned, e.g., numbers. There is nothing inherent in the number 3 to indicate what it represents. So, it must be culturally learned.

By adopting the Peirce theory of signs, we suggest that the notation designer creates a visual representation by following the order of visual representation type as follows:

- (a) Create a visual representation that can represent the semantics of the element (semantic transparency) by using an **icon-type** visual representation. For example, an icon of Stickman figure as commonly used to represent the semantic of an Actor such as in a UML Use Case diagram as illustrated in Figure 3.17.
- (b) If there is no suitable icon that can be used to convey the semantic of the element then create a visual representation that can provide a clue to the (majority) of the targeted notation users by using the **index-type** visual representation. For example, adopting an

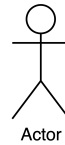


FIGURE 3.17: A Stick-man icon represents an Actor in UML

existing visual representation to visually represent a particular element in order to provide a clue to the notation user who is familiar with the existing visual representation, as long as these elements have identical semantics.

Lets consider the visual representation of the UML Package as illustrated in Figure 3.18 as an example of index-type visual representation.

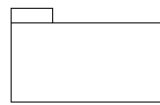


FIGURE 3.18: The UML Package

The visual representation of the UML Package has been adopted to represent the Package element of the SysML notation. Therefore, the visual representation of the SysML package (as shown in Figure 3.19) is identical to the visual representation of the UML Package. By adopting the UML Package visual representation, a user of the SysML (who is familiar with UML notation) may easily understand the meaning of the visual representation.



FIGURE 3.19: The SysML Package

It is important to notice that using the existing visual representation to visually represent the semantic of an element can be applied only when the semantics of the elements are identical. The notation user could be misunderstand the meaning of the element if this element as an existing visual representation of another element that has different semantics.

- (c) If there is still no index-type that can be used to represent the semantic of the element then create a **symbol-type** visual representation to be learned by the notation users. For example, the visual representation of I* Agent that is visually represented using a circle with a horizontal line placed on the upper part of the circle (as illustrated in Figure 3.20).

An I* Agent is defined as an actor with concrete, physical manifestations, such as a human individual, an organization, or a department. For example: Travel agency, Student, and University of Wonderland [18]. The visual representation of I* Agent does not represent

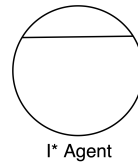


FIGURE 3.20: I* Agent visual representation

the meaning of its semantics, instead it needs to be learned by the notation users to get familiar with the I* Agent visual representation and its corresponding semantics.

Figure 3.21 illustrates the degree of the semantic transparency of the visual representation type. An icon represents more literal visual representation since it can represent the semantic of the element, and a symbol representing more abstract visual representation since it is not directly representing the semantic of the element and also it needs to be learned by the notation user in order to understand the semantics that is being represented by the symbol.



FIGURE 3.21: Semantic transparency of the visual representation types

Defining which visual representation type to be used to visually represented an element, can help the notation designer in documenting the rationale behind the development of the visual representation. In order to create a visual representation, for any type of visual representation, we can utilise and combine visual variables.

3. Perceptual discriminability

Perceptual discriminability is defined as the creation of a visual representation that must be distinguishable from other visual representations. Distinguishable visual representations can help notation users identifying different visual representations [3, 15, 22]. Therefore, this principle is suggested to be considered by the notation designer to create visual representations.

Discriminability can be measured by the *visual distance* between visual representations. The visual distance can be determined by identifying the different visual variable values used by the visual representations. If visual representations use exactly the same visual variables then these visual representations are identical which means the visual distance between these visual representations is equal to zero. Lets consider an example as shown in Figure 3.22 to illustrates visual distance between visual representations.

A De Marco Data Flow Diagram (DFD) is visually represented using a visual variable Shape with value Rectangle. Meanwhile, the visual representation of a Process uses the visual variable

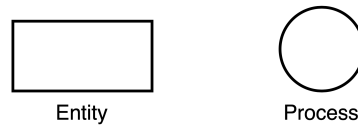


FIGURE 3.22: Visual distance between visual representations of De Marco DFDs Entity and Process

Shape with value Circle. Therefore, these visual representations (as shown in Figure 3.22) can be distinguished because they utilised different visual variable value, or in other words the visual distance between them is not zero.

4. Redundant coding

Redundant Coding is defined as part of the Perceptual Discriminability principle in [3]. It is a technique to increase the perceptual discriminability between visual representations by using multiple visual variables (and their unique value) in creating visual representations. Figure 3.23 illustrates a visual representation that applies redundant coding.



FIGURE 3.23: Redundant coding used in UCM Actor visual representation

The Use Case Map (UCM) Actor, as shown in Figure 3.23, applied redundant coding in its visual representation [58, 80]. There are several visual variable values used to compose the UCM Actor such as a Rectangle of visual variable Shape; and a Stick-man icon that is placed on the top-left of the rectangle (Position/Location visual variable).

Applying redundant coding could increase perceptual discriminability, however, the excessive use of visual variables may cause a negative impact on visual representation usability. Further explanation regarding this aspect is discussed in the Notation Usability sub-section along with some examples.

3.4.4 Identify notation usability

In creating visual representations, usability factors need to be considered by the notation designer in order to enable a good user experience, and user satisfaction in using the notation [48, 81]. The following usability factors are suggested as part of our approach that can be considered in designing a visual notation:

1. Ease of drawing

In some cases models are developed in an interactive manner between developer and other stakeholders by sketching on whiteboards or paper. Sketching a model using a notation for

some developers can be considered as an integral activity to informally share knowledge related to the system to another stakeholders [82].

Since sketches are often hand-drawn, written on a paper or whiteboard without any additional technology tool, the visual representations must be easy to draw so the modellers can easily use the notation to create/sketch a model [48]. Therefore, the visual representations that are used to create the diagrams should be design as simple as possible so developer can draw the diagram quickly and easily and do not slow down the flow of ideas.

By considering the easy to draw aspect of a visual representation, visual variables such as Colour can be difficult to use due to limitation in drawing ability and availability of drawing equipment such as colour pens. Moreover, Colour is sensitive to variations in visual perception such as color blindness and screen or printer characteristics e.g., black-and-white printers. We also suggest to avoid creating a complex visual representation such as designing a visual representation by excessively combining visual variables and/or using too many different values of a visual variables, since it could require more effort for the modeller to draw the visual representation. We consider a visual representation as illustrated in Figure 3.24 as an example of a complex visual representation.

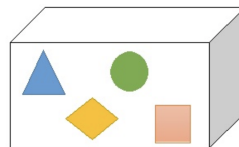


FIGURE 3.24: Example of a complex visual representation (in context of a hand-drawn visual representation)

The visual representation that is shown in Figure 3.24 is a user-created visual representation for representing a UML Package concept as a result of an experiment on the investigation of semantic transparency of the UML notation as reported in [51]. We consider the user-created visual representation as shown in Figure 3.24 to be a complex visual representation since it uses excessive different types of visual variable Shape and Colour in a single visual representation, such as Triangle, Circle, Diamond, and Square (for the 2D shape), and also the Cuboid shape (for the 3D shape type), where each of them has different types of Colour. The modeller might spend more effort and time to draw the different types of Shape and Colour as well as the Positioning of the Shapes when creating/sketching a model by hand or without a (technological) tool support.

According to the literature as stated in [3], redundantly using visual variables (redundant coding) can help to increase *perceptual discriminability* factor (different visual representations should be clearly distinguishable from each other) and allows positive effects such as a faster

detection of modeling constructs. However, the excessive use of different types of visual variables and values also could result in a negative impact to the easy to draw factor since the modellers may need to spend more effort to draw the visual representation.

In contrast to the visual representation of user-created visual representation of UML Package as illustrated in Figure 3.24, we consider the original visual representation of UML Package as relatively easy to draw. In UML, a Package is rendered as a tabbed folder - a rectangle with a small tab (another rectangle) attached to the left side of the top of the rectangle as illustrated in Figure 3.18.

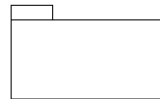


FIGURE 3.25: The original UML Package visual representation

The design of the original UML Package, as illustrated in Figure 3.25, may not as stand out as the design of the user-created UML Package as illustrated in Figure 3.24. However, the modeller might spend less effort and time in drawing the visual representation of the original UML Package than drawing the user-created UML Package.

In visual notation design, the term easy to draw can be subjective and highly depend on, for example, the skill of the modeller in drawing a particular visual representation. Therefore, in order to reduce subjectivity, we suggest the visual notation designer, after considering the easy to draw factor in designing a visual representation, to assess or evaluate the created visual representation. For example, by asking the targeted notation user to actually use or draw the created visual representation in the context of developing a model using the proposed notation and analyse their performance in using the notation as well as seek their feedback.

Beside the subjectivity aspect and the interaction between the easy to draw and the perceptual discriminability factor, it is also important to notice that there is also an interaction between the easy to draw factor and the intuitiveness of a visual representation. In this context, considering the easy to draw factor in designing a visual representation may influence negatively the intuitive design of a visual representation.

A visual representation should be designed as intuitive as possible in order to help the user instantly understand the meaning of the visual representation only by looking at the appearances of the visual representation. This idea is also known as the Semantic Transparency of a visual representation [3].

In some cases, just like considering the perceptual discriminability factor, an intuitive visual representation may require more effort to draw. For example, consider an example of visual representation of the Compromised State in an extension of UML statecharts notation to model

security aspects. In this context, Compromised State is used to indicate that the system has been compromised. It indicates that damage has occurred and is still occurring. Figure 3.26 illustrates the original visual representation of the Compromised State as discussed in [6].



FIGURE 3.26: The visual representation of Compromised State

In order to create an intuitive visual representation of the Compromised State, a proposal for an improvement to the original visual representation of the Compromised State is made. As reported in [6], the author proposed an alternative notation to the original visual representation of the Compromised State by adding the colour black as the background of the visual representation and an icon of skull and bones. The colour black is used since many existing security modeling languages used this colour to indicate misuse, including misuse cases, and mal-activity diagrams. Based on this rationale, the color black is used by the author as the background in the new visual representation of the Compromised State. A skull and bones sign is used to indicate that damage has already occurred or is currently occurring. Based on the experimental study conducted by the author, the results shows that the new visual representation of the Compromised State is more intuitive than the original one. However, in the context of easy to draw (e.g. hand-drawing the model), the original notation is relatively easy to draw and requires less effort compared to the new visual representation illustrated in Figure 3.27.



FIGURE 3.27: Alternative Compromised State visual representation as suggested in [6]

Knowledge of interactions such as between the easy to draw and the semantic transparency of a visual representation can be used to make trade-offs (where a factor conflicts with another factor) and analysis of the design rationale of a visual representation.

2. The use of Colour

Visual notation elements (constructs) are mainly distinguished on the basis of the visual variable Shape because it has a significant impact on object recognition [3, 77]. In order to achieve faster detection of a visual representation and distinguish between one visual representation and another, *redundant coding* technique in designing a visual representation is suggested in visual notation design studies such as in [3, 15, 17, 24]. Redundant coding is a technique of using multiple visual variables for designing a visual representation in order to increase perceptual discriminability between visual representations (distinguishing between visual representations).

Just like other visual variables, Colour is a visual variable that can be used to implement dual coding in notation design. Colour has been tested empirically for conceptual modelling such as in [7, 8, 83], that shows if Colour is used as an additional variable to emphasise a particular aspect in conceptual modelling, significant positive effects on comprehension for users without prior knowledge of modelling language can be achieved.

An example of the usage of Colour in the context of redundant coding in visual modelling language can be found in [7] where Colour was discussed to visually distinguish matching operators in Business Process Models (BPMs), as illustrated in Figure 3.28.

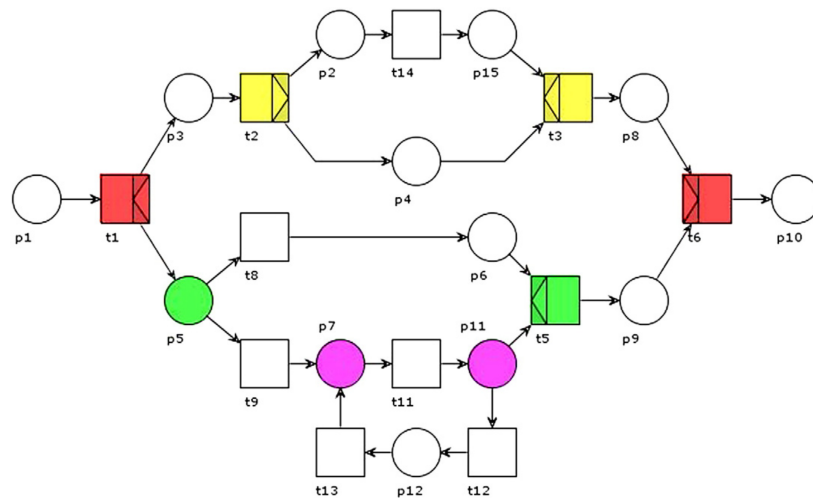


FIGURE 3.28: The use of Colour as a syntax highlighting technique [7]

As shown in Figure 3.28, Colours are used as the syntax highlighting technique in order to give emphasis to particular aspects in the created model. For example the red colour is used to emphasise the semantics of an AND-split/AND-join pair as visualised in element t1 and t6, and the magenta colour is used to emphasise the semantics of an implicit XOR-split/implicit XOR-join pair as visualised in element p11 and p7. The use of syntax highlighting technique has become an established feature in programming editors to support programmers in making sense of code. This idea has been adopted in visual modelling language, for example, as implemented in [7].

Another example of the use of Colour in visual modelling language can be found in [8] where Colour is used as part of the Opacity-Driven Graphical Highlight technique for increasing the cognitive effectiveness of Business Process Diagrams (BPDs) by changing the opacity of graphical elements. To support the implementation of the Opacity-Driven Graphical Highlight technique, the authors developed a prototype of a supporting tool where Colour is used partially to implement opacity of graphical elements as illustrated in Figure 3.29.

Based on the experiment related to the effectiveness of the Opacity-Driven Graphical Highlight

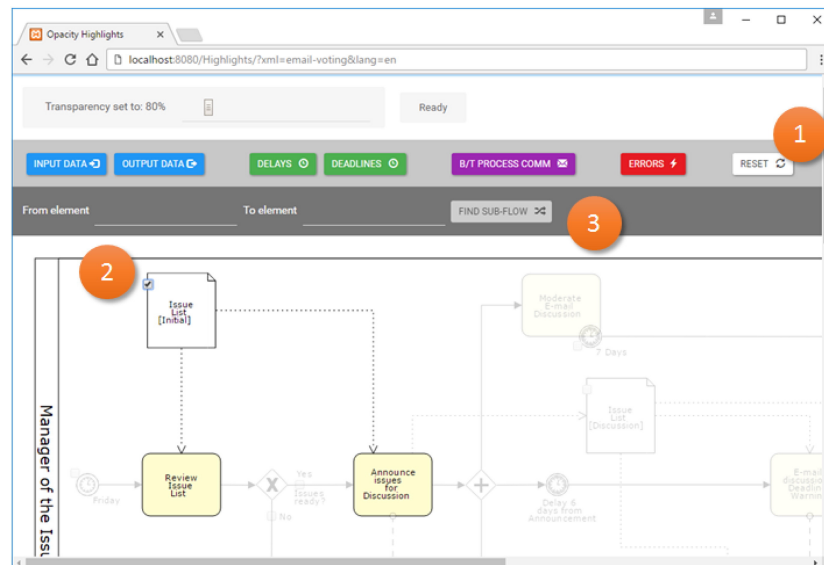


FIGURE 3.29: The use of Colour as part of the Opacity-Driven Graphical Highlight technique [8]

technique, the authors in [8] claimed that participants of the experiment that used Opacity-Driven Graphical Highlights significantly outperformed those that used the conventional approach (i.e. process modeling tools which does not exploit the opacity of Business Process Diagrams elements).

The authors also conclude that using Opacity-Driven Graphical Highlights increases the cognitive effectiveness of business process diagrams [8]. However, the Opacity-Driven Graphical Highlight technique highly depends on the development of a model using tool support and will be hard to implement for users who create a model by hand-drawing.

Despite the benefit of the use of Colour to increase perceptual discriminability between visual representations, it is important to notice that color should never be used as the main variable and the sole basis for distinguishing between visual representations as it is sensitive to variations in visual perception such as color blindness and screen or printer characteristics.

We use Figure 3.28 to illustrate an example of the limitation of colour in the context of sensitivity to variation such as printer characteristics, specifically, in the case of printing the model using a black-and-white printer.

The Colours that are supposed to denote specific information as shown in Figure 3.28 cannot be seen clearly when the model is printed using the black-and-white printer (as illustrated in Figure 3.30). In such cases, the notation user might miss some crucial information that they were supposed to received. For example, the red colour that was used to emphasise the semantics of an AND-split/AND-join pair, as visualised in element t1 and t6, cannot be seen clearly since the model was printed in black-and-white mode.

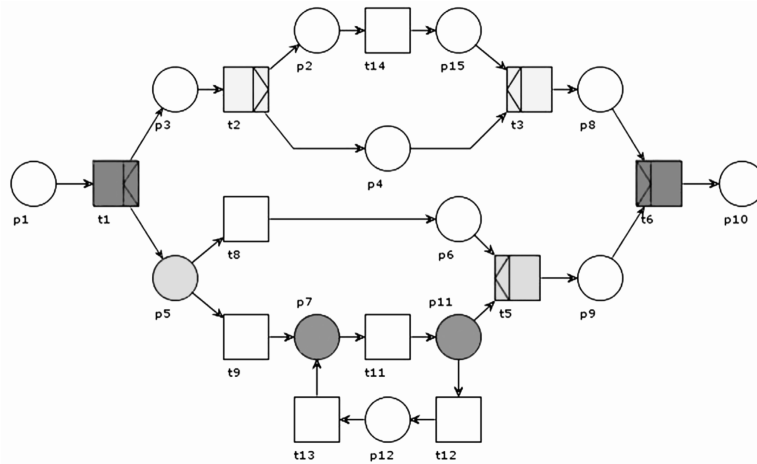


FIGURE 3.30: Example of a model that used Colours as a syntax highlighting technique but was printed using a black-and-white printer

3. Tool support implementation

In some cases, models are developed by sketching on whiteboards or paper by developers. In other cases, some developers rely on a software tool when performing their model development activities [84]. Considering the applicability of the visual representations in tool support implementations should be considered by the notation designer because there are developers who use software tools to create models. On the other hand, improving Semantic Transparency and Perceptual Discriminability may lead to a notation that is harder to draw or sketch by hand. However, many modeling activities are performed via tool support and thus the new notation would benefit from the support provided by tools when they are available.

Previously an example of tool support was provided as illustrated in Figure 3.29 to facilitate the notation user of the Business Process Diagrams that implements the Opacity-Driven Graphical Highlight technique. Another example of tool support implementation is the Artoo Argumentation Tool that is developed by researches at the University of York. Figure 3.31 illustrates a screenshot of the Artoo argumentation tool as adopted from [5].

Developing tool support like Artoo Argumentation Tool that implements the GSN-like notation can be used as an evaluation that the created notation is feasible when implemented with tool support.

3.5 Step 3: Consider visual inheritance and create visual representation

The next step is to create the visual representations. This step requires the output from Step 1 (the created design path) and Step 2 (identified design constraints) as the input. Figure 3.32 illustrates the input, process, and output related to this step.

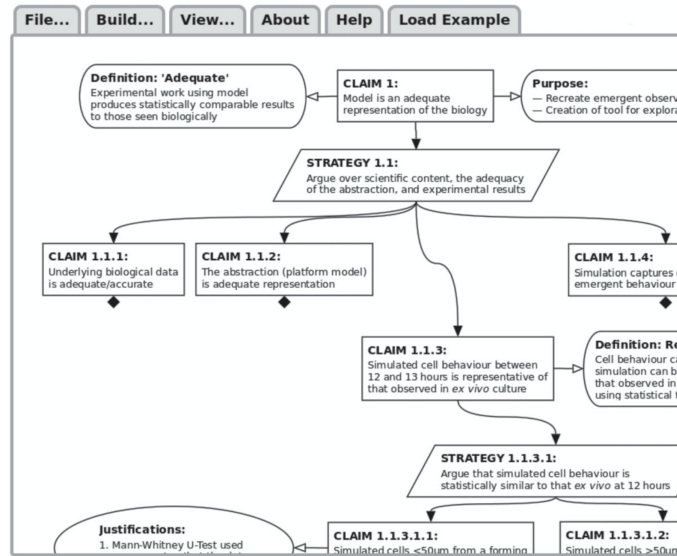


FIGURE 3.31: A screenshot of the Artoo argumentation tool [5]

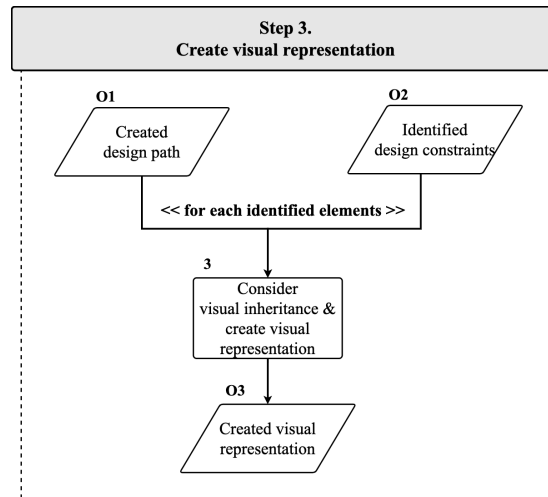


FIGURE 3.32: Step 3: Create visual representation

We need to create the visual representation of each element that has been identified in Step 1 (Ou.1), and to create the visual representation, we need to respect the design constraints as identified and considered in Step 2 (Ou.2) and visual inheritance from other related elements. For each identified element, the process of considering design constraints, visual inheritance and the creation of the visual representation are described in detail in this section. The expected output of this step is the created visual representation (Ou.3).

The focus of this step is to create the visual representation of the identified elements. To create the visual representation we can utilise the visual variables and consider the defined design constraints. In our approach, it is possible to create similar visual representations for two or more related elements through the visual inheritance process. Visual inheritance is the process of inheriting the design constraints that are used to develop a visual representation of an element to another related element

for the purpose of creating the visual representation of the element that received the inherited design constraints.

The idea of visual inheritance is commonly applied in the field of User Interface design to maximise design effectiveness and efficiency [70, 71, 85]. Although the idea of visual inheritance is commonly applied in the field such as User Interface design, this idea is a lack of systematic application in the field of visual notation design. In this thesis, we attempt to propose the application of the visual inheritance in visual notation design as part of the our visual notation design approach.

In our approach, visual inheritance is important in order to support family resemblance among visual representations. Metamodel elements that have similar semantics should be visualised using similar visual representations. In other words, the visual representation similarity should be used to indicate semantic similarity between elements: family resemblances among visual representations can be exploited to show a family relationship among elements [3, 86].

To create similar visual representations for elements that have similar semantics, we need to identify which elements have similar semantics. In other words, by identifying the elements that have similar semantics, we are identifying the applicability of the visual inheritance to create the visual representation of the elements.

In UML, for example, we can identify two or more metamodel elements that have similar semantics by identifying two or more Classes that are connected using the Generalisation relationship. The Generalisation relationship is a relationship between a more general Class (super Class) and more specific Class (sub Class). If there are Classes that are connected using generalisation relationship, they may have similar semantics. We need to understand the meaning of the semantics of these connected Classes in order to make sure if they have similar semantics.

Let's consider Figure 3.33 that illustrates the example of three Classes that are connected using a Generalisation relationship.

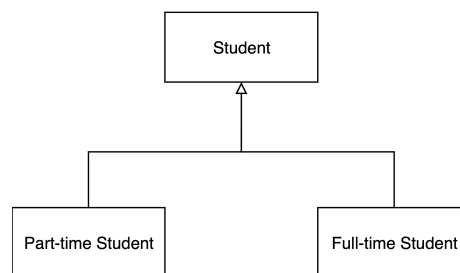


FIGURE 3.33: Example of Generalisation relationship

Suppose we have a Class named Student in this example. This class is the super-Class (i.e. more general class) for the other Classes, namely Part-time Student Class, and Full-time Student Class.

Suppose the semantics of the Student Class, in the context of undergraduate study in a university, is defined as:

"A person who is studying at a university or college"

The semantics of the Full-time Student Class is defined as:

"Full-time undergraduate students are defined as those taking nine credits or more per academic term"

The semantics of the Part-time Student Class is defined as:

"Part-time undergraduate students are defined as those taking fewer than nine credits per academic term"

Using the semantics of the elements of this example, we may conclude that three of these classes have semantic similarity since they mainly define a type of person who studies at a university or college, and the sub-classes of the Student Class (i.e. Part-time and Full-time Student) define a more specific type of a Student.

In the case, that we have two or more elements that have similar semantics, we can create similar visual representations for them by utilising the visual variables and their values, specifically, by inheriting these to the visual representation for the other element. Where this is the case, the visual representation that inherits the visual variables must make a modification of the inherited visual variables or their values to represent the semantics of the element. This will avoid identical visual representations (later in this section, we describes the importance of avoiding identical visual representation on visual notation design).

By considering the applicability of the visual inheritance in creating the visual representations of the related elements, we can show family resemblance between created visual representations.

Let us reuse the example in Figure 3.33 to illustrate the process of creating similar visual representations for elements that have similar semantics. As the first step, we need to create a design path, and then we need to decide which visual variables to use to create visual representations as well as consider other design constraints.

Let us call the example in Figure 3.33 the Student Metamodel. Previously, we studied this metamodel by understanding the semantics of the classes as well as the relationship between classes. Suppose we define the design path of the Student Metamodel as illustrated in Figure 3.34. The design path starts from the Student class (as the root class) then continue to the Full-time Student class, and is followed by the Part-time Student class. In this case, after creating the visual representation of the Student class, we continue to create the visual representation of the Full-time Student class before

the Part-time Student class due to, for example, consideration of the semantic importance between these classes. The Rectangles in the metamodel are representing the Classes, the Line with a hollow arrowhead that connects the classes represents the Generalisation relationship, and the arrowhead with the dashed lines represents the design path.

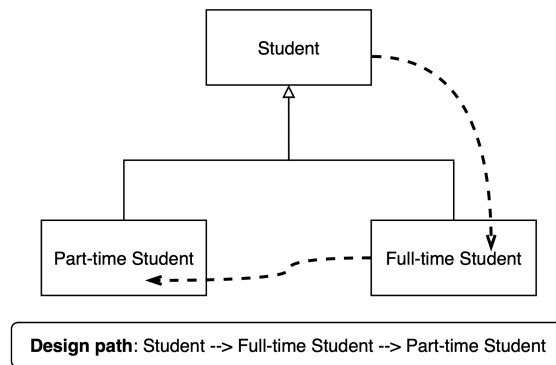


FIGURE 3.34: Example of a design path of the Student Metamodel

After a design path is created, we need to define which visual variables will be used to design the visual representations of the Student metamodel elements and the other design constraints. Suppose, in this example, we decide to use the following visual variables: Shape, Position/Location, Brightness, Size, and Texture.

The other visual variables such as Colour and Orientation are not being used in this example due to considering the Usability aspect of the visual representation and also providing free visual variables to be used, for example, by other notation designers in order to personalise or adopt the visual representations (e.g. the use of visual variable Orientation in the implementation of the Artoo Argumentation Tool).

After deciding which visual variables to use, we need to decide the other design constraints that need to be respected in creating the visual representations. In this example, we adopt the following design constraints:

- Avoiding clash with existing notations due to reduce misinterpretation for some notation users who may be familiar with existing notations.
- Adopting existing visual notation design principles: Semiotic clarity, Semantic transparency, Perceptual discriminability, and Redundant coding.
- Notation usability

Adopted to provide good user experience in using the visual representations by creating:

- Easy to draw visual representation such as create a simple visual representation considering some notation users sketch or hand-draw the diagram or model on whiteboards or

paper.

- Avoid the use of Colour considering the different visual perception such as colour blindness and printer characteristics (e.g. black-and-white printers).
- The creation of the visual representation needed to consider the feasibility of the application in tool support.

The decided visual variables and the design constraints need to be respected in creating the visual representation of the Student metamodel elements. When applicable (i.e. in the case there are visual representations that have similar semantics), all the visual variables that are available to be used to create the visual representations, as well as the other design constraints, can be inherited in designing visual representations that have similar semantics.

Suppose we visually represent the Student Class using a Stick-man figure, as shown in Figure 3.35.

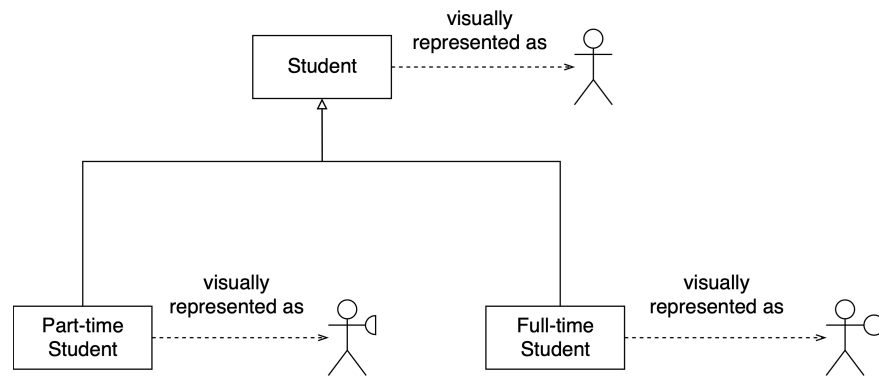


FIGURE 3.35: Example of the visual inheritance

The visual representation of the Stick-man figure as illustrated in Figure 3.35 is composed of the following visual variables:

- Shape: 'Circle' and 'Lines'
- Brightness: 'Light' (as background within the circle)
- Size: '1 pt' (thickness of the shapes)
- Position/Location: 'Circle is located on the upper-part of the combined lines' and lines combined together forming a stickman figure as illustrated in Figure 3.35.
- Texture: 'Solid line' used as the line texture type of the shapes.

We considered the following design constraints in the usage of the Stick-man figure to represent the Student class element:

- Consider existing notations: (for example) no existing notation in the same domain uses the Stick-man figure to represent a Student.
- Consider existing notation design principles:
 - Semiotic clarity: the Stick-man figure is used only for representing the Student class.
 - Semantic transparency: the Stick-man figure is used as an icon-type visual representation to suggest the semantic meaning of the Student class.
 - Perceptual discriminability: the Stick-man figure is used only to represent the Student class element.
 - Redundant Coding: the Stick-man figure is created using multiple visual variables and different values.
- Consider notation usability:
 - Easy to draw: we considered the Stick-man figure is relatively easy to draw (notation users evaluation is needed to reduce subjectivity).
 - The use of Colour: not using Colour to create the Stick-man figure.
 - Tool support implementation: we considered the Stick-man figure feasible to implement in tool support (further evaluation is needed to reduce subjectivity).

Since the sub-classes of the Student Class have semantic similarity to the Student Class, we can create similar visual representations for the sub-classes (i.e. Part-time and Full-time Student Class) and the super-class (i.e. Student Class). We can inherit the visual representation of the Student Class to its sub-classes. The inherited visual representation needs to be modified based on the semantics of each sub-class. For example, Full-time Student Class, in order to represent its specific semantic, we modify the inherited visual representation of the Student class (i.e. Stick-man symbol) by adding other visual variables to the inherited visual representation as follow:

- Shape: 'circle'.
- Brightness: 'Light' (as background within the circle).
- Size: '1 pt' (thickness of the lines of the shape).
- Position/Location: 'Right-side of the Stick-man figure' for the placement of the 'circle'.
- Texture: 'Solid line' used as the line texture type of the shapes.

The newly added visual variables are used to denote the semantics of 'Full-time Student'. Therefore, the Full-time Student class uses the following visual variables including the inherited visual variables:

- Shape:
 - 'Circle' (inherited).
 - 'Lines' (inherited).
 - 'Circle'.
- Brightness:
 - 'Light' as background within the circle (inherited).
 - 'Light' as background within the other circle.
- Size:
 - '1 pt', thickness of the lines of the shape (inherited).
 - '1 pt', thickness of the lines of the shape of the other circle.
- Position/Location:
 - 'Circle is located on the upper-part of the combined lines' (inherited).
 - 'Lines combined together forming a Stick-man figure' (inherited).
 - 'The other circle is located on the right-side of the Stick-man figure' as illustrated in Figure 3.35.
- Texture:
 - 'Solid line' used as the line texture type of the shapes (inherited).
 - 'Solid line' used as the line texture type of the shape of the other circle.

The newly added Circle that is placed on the right-side of the Stick-man figure is used to represent the specific meaning of the Full-time Student class. This visual representation also considers and respects the inherited design constraints as follow:

- Consider existing notations: (for example)

- no existing notation in the same domain uses the Stick-man figure to represent a Student (inherited).
- no existing notation in the same domain uses a circle placed on the right-side of the Stick-man figure to represent a Full-time Student.
- Consider existing notation design principles:
 - Semiotic clarity:
 - * The Stick-man figure is used only to represent the Student Class (inherited).
 - * The Circle placed on the right-side of the Stick-man figure is used only to represent the Full-time Student class.
 - Semantic transparency:
 - * Stick-man figure is used as an icon-type visual representation to suggest the semantic meaning of the Student class (inherited).
 - * The circle that is placed on the right-side of the Stick-man figure is used as an index-type visual representation to suggest the meaning of the ‘Full-time’.
 - Perceptual discriminability: The visual representation of the Student class and Full-time Student class can be distinguished by the newly added circle that is placed on the right side of the Stick-man. Hence, the visual distance between these visual representations is not equal to zero.
 - Redundant Coding:
 - * Stick-man figure is created using multiple visual variables and different values (inherited).
 - * The circle that is placed on the right-side of the Stick-man figure is created using multiple visual variables and different values.
- Consider notation usability:
 - Easy to draw:
 - * We considered the Stick-man figure is relatively easy to draw (notation users evaluation is needed to reduce subjectivity). (inherited).
 - * We considered that the Circle placed on the right-side of the Stick-man figure is relatively easy to draw (notation users evaluation is needed to reduce subjectivity).

- The use of Colour:
 - * Not using Colour to create the Stick-man figure (inherited).
 - * Not using visual variables Colour and Orientation to create the Circle.
- Tool support implementation:
 - * We considered the Stick-man figure feasible to implement in tool support (further evaluation is needed to reduce subjectivity). (inherited).
 - * We considered the Circle placed on the right-side of the Stick-man figure feasible to be implemented in tool support (further evaluation is needed to reduce subjectivity).

As for the Part-time Class, it inherits the Stick-man figure to represent the semantics of a Student (general representation of the semantic of a Student). The Part-time Class has specific semantics to indicate that the part-time student is 'a student who takes fewer than nine credits per academic term'. In this example, we modified the Stick-man figure by adding other visual variables to the visual representation as follow:

- Shape: 'Half-circle'.
- Brightness: 'Light' (as background within the half-circle).
- Size: '1 pt' (thickness of the lines of the shape).
- Position/Location: 'Right-side of the Stick-man figure' for the placement of the 'Half-circle'.
- Texture: 'Solid line' used as the line texture type of the shapes.

The newly added visual variables are used to denote the semantic of 'Part-time Student'. Therefore, the Part-time Student class used the following visual variables, including the inherited visual variables:

- Shape:
 - 'Circle' (inherited).
 - 'Lines' (inherited).
 - 'Half-circle'.
- Brightness:
 - 'Light' as background within the circle (inherited).

- ‘Light’ as background within the half-circle.
- Size:
 - ‘1 pt’, thickness of the lines of the shape (inherited).
 - ‘1 pt’, thickness of the lines of the shape of the half-circle.
- Position/Location:
 - ‘Circle is located on the upper-part of the combined lines’ (inherited).
 - ‘Lines combined together forming a Stick-man figure’ (inherited).
 - ‘Half-circle is located on the right-side of the Stick-man figure’ as illustrated in Figure 3.35.
- Texture:
 - ‘Solid line’ used as the line texture type of the shapes (inherited).
 - ‘Solid line’ used as the line texture type of the shape of the half-circle.

The Half-circle is used to represent the specific meaning of the Part-time Student class. This visual representation also considers and respect the inherited design constraints as follow:

- Consider existing notation: (for example)
 - no existing notation in the same domain uses the Stick-man figure to represent a Student (inherited).
 - no existing notation in the same domain uses the half-circle placed on the right-side of the Stick-man figure to represent a Part-time Student.
- Consider existing notation design principles:
 - Semiotic clarity:
 - * The Stick-man figure is used only to represent the Student Class (inherited).
 - * The half-circle placed on the right-side of the Stick-man figure is only used to represent the Part-time Student class.
 - Semantic transparency:

- * Stick-man figure is used as an icon-type visual representation to suggest the semantic meaning of the Student class (inherited).
- * Half-circle is used as an index-type visual representation to suggest the meaning of the 'part-time'.
- Perceptual discriminability: The visual representation of the Student class, Full-time Student class, and the Part-time Student class can be distinguished by the newly added half-circle that is placed on the right side of the Stick-man (of the Part-time Student class visual representation). Hence, the visual distance between these visual representations is not equal to zero.
- Redundant Coding:
 - * Stick-man figure is created using multiple visual variables and different values (inherited).
 - * Half-circle is created using multiple visual variables and different values.
- Consider notation usability:
 - Easy to draw:
 - * We considered the Stick-man figure is relatively easy to draw (notation users evaluation is needed to reduce subjectivity). (inherited).
 - * We considered the Half-circle is relatively easy to draw (notation users evaluation is needed to reduce subjectivity).
 - The use of Colour:
 - * Not using Colour to create the Stick-man figure (inherited).
 - * not using visual variables Colour and Orientation to create the half-circle.
 - Tool support implementation:
 - * We considered the Stick-man figure feasible to be implemented in tool support (further evaluation is needed to reduce subjectivity) (inherited).
 - * We considered the Half-circle feasible to be implemented in tool support (further evaluation is needed to reduce subjectivity).

All of the visual variables and the other design constraints used to visually represent the Student Class is inherited to its sub-classes (i.e. Part-time Student and Full-time Student Class) in order to create a

similar visual representation of these classes and to show visual representations resemblance.

The creation of all the visual representations are considered easy to draw, e.g., Colour has not been used in designing the visual representations. Further evaluation to reduce subjectivity related to the easy to draw aspect can be done, for example, by conducting an empirical study involving the notation users to assess that the created visual representations are easy to draw (e.g. in a model creation task simulation).

To summarise, a visual representation can be created by utilising and combining visual variables. The other design constraints such as consider existing notations, existing notation design principles, and notation usability can be considered in the creation of the visual representations.

The applicability of the visual inheritance can be identified by understanding the semantics of the metamodel elements, and in the case where there is a semantic similarity between elements (e.g. can be identified by a generalisation relationship between elements) then it is possible to apply the visual inheritance process in creating similar visual representations between semantically related elements. It is important to be noticed that, the created visual representations of two or more elements must not be identical in order to avoid visual representations overload.

In the next section, the documentation of the design rationale of the visual representations is described.

3.6 Step 4: Document design rationale

As previously mentioned in Section 3.2 related to our view regarding the life cycle of a visual notation, we see the visual notation development as an iterative process. Therefore, it is important to document the design rationale of each created visual representation as part of the visual representation identity that can be used as an input for the improvement of the notation, for example, for further iteration of the notation development.

Figure 3.36 illustrates the input, process and output in this step, where the created visual representation (and its rationale in the design process) is used as the input in documenting the design rationale. The expected output from this process is the documented design rationale.

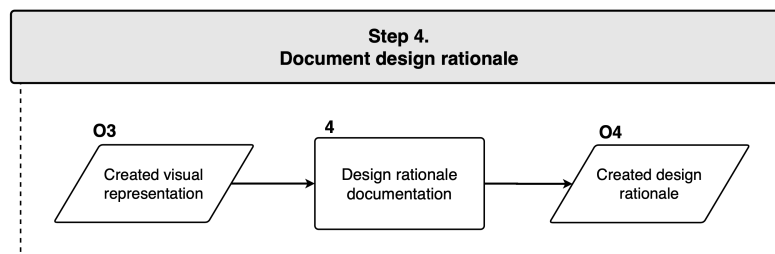


FIGURE 3.36: Step 4: Design rationale documentation

The documentation of design rationale can be seen as part of the documentation of the visual representation identity that can be created by describing the name of the element, the visual representation, the semantics, the visual variables used, the information regarding the considered visual inheritance in creating the visual representation, and the design rationale such as the design constraints adopted in the creation of the visual representation. The compositional rules of the visual representation, when used in a diagram, is also important to be documented. Table 3.1 shows an example of a template that can be used to document a visual representation identity.

TABLE 3.1: Visual representation identity template

Visual representation identify of		<element name >
1	Visual representation	
2	Semantics	
3	Visual variable used	
4	Visual inheritance info	
5	Design rationale (design constraints adopted)	
		Avoiding clash with existing notation
		Adopting existing visual notation design principles
		Semiotic clarity
		Semantic transparency
		Perceptual discriminability
		Redundant coding
		Notation Usability
		Easy to draw
		Use of colour
	Tool support implementation	
	Other constraints (if any)	
6	Compositional rules when used in a diagram	

The Student Class from the Student Metamodel example is used to illustrate the visual representation identify example, as shown in Table 3.2.

TABLE 3.2: Student visual representation identity

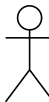
Visual representation identify of		Student
1	Visual representation	
2	Semantics	A person who is studying at a university or college
3	Visual variable used	Shape: 'Circle' and 'lines'.
		Brightness: 'Light' as background within the circle.
		Size: '1 pt' (thickness of the shapes).
		Position/Location: 'Circle is located on the upper-part of the combined lines' and 'lines combined together forming a stickman figure'.
	Texture: 'Solid line' used as the line texture type of the shapes.	
4	Visual inheritance info	Not inherited any visual representation from other elements
	Design rationale	
	Avoiding clash with the existing notations	There is no existing notation in the domain that uses a Stick-man figure to represent the semantic of a Student.
	Adopting existing visual notation design principles	

Table 3.2 continued from previous page

Visual representation identify of		Student
	Semiotic clarity	The Stick-man figure is used only to represent the Student class.
	Semantic transparency	An icon of the Stick-man figure is used to facilitate the semantic transparency of the Student class.
	Perceptual discriminability	The visual distance between visual representation of the Student class and other visual representations is not equal to zero. Hence, no visual representation redundancy occurred.
	Redundant coding	The Stick-man figure is created using multiple visual variables and different values.
	Notation Usability	
	Easy to draw	We considered the Stick-man figure is relatively easy to draw (notation users evaluation is needed to reduce subjectivity).
	Use of colour	Not using Colour to create the Stick-man figure.
	Tool support implementation	We considered the Stick-man figure feasible to be implemented in tool support (further evaluation is needed to reduce subjectivity).
	Other constraints (if any)	N/A.
6	Compositional rules when used in a diagram	(For example) The Stick-man figure can be connected to other entity via: PartOf relationship and Owned relationship

The visual representation identity including the design rationale is the expected output of this step. In order to complete the visual notation design process, we need to do an iterative process of creating visual representation and documenting its rationale until all the identified elements have their visual representation.

3.7 Summary

In this chapter, we described an approach that we developed for creating a visual notation from a defined metamodel as the main input.

The proposed approach is developed to reduce the gap in the visual notation design field in which there is currently a lack of systematic steps that can be adopted in the creation of a visual notation that considers the hierarchy of a defined metamodel as the main input. This approach also can be seen as a contribution in the field of visual notation design studies.

The general idea of proposed visual notation design approach is to create an effective and efficient notation that considered the hierarchical structure of the elements from a defined metamodel where several design constraints are also considered as part of the process, for example, considering existing notations, existing notation design principles, and the notation usability factors. The metamodel elements including its hierarchical structure and the considered design constraints are used as an input to create the visual representation of the metamodel elements.

Related to the creation of the visual representation of the metamodel elements, a visual inheritance process is introduced as an approach to create similar visual representations for elements that are related each others in terms of their semantics (i.e. semantics similarity). Creating similar visual representations for semantically related (similar) elements is important in order to show family resemblance among related visual representations (of elements that have similar semantics) that might help notation user to identify and learn about the visual representations of the elements.

After the visual representations of the metamodel elements are created, as part of our approach, we suggest that the notation designer documents the design rationale for each created visual representation that can be used as an input for the improvement of the notation, for example, for further iteration of the notation development.

The expected result of the application of the proposed visual notation design approach is an effective notation that can be evaluated by the notation stakeholders such as the notation user.

In the next section, the application of the proposed visual notation design approach is described in the context of the SACM argumentation notation development.

Chapter 4

The development of the SACM notation

4.1 Introduction

In Chapter 1, we described one of the objectives of this thesis: to propose and develop, as an alternative representation for the textual form, an effective visual notation for the SACM metamodel since no visual notation is provided in the SACM specification version 2.0 [9].

In order to create the notation for the SACM metamodel, we developed the visual notation design approach described in Chapter 3. The design approach was created due to the lack of an existing systematic approach that can be adopted for the development of a notation based on a defined metamodel. The proposed visual notation design approach is one of the contributions of this thesis along with the SACM notation presented in this chapter.

The SACM metamodel specification version 2.0 is used as the primary input for the visual notation design process. The expected output is an effective notation that is accurate, easy to learn, and easy to use by its users.

In the next section, the development of the SACM notation using the proposed visual notation design approach is described. The process starts with the preparation steps of defining the design scope and considering the involvement of the notation users. After that, the design process is conducted by following the 4 steps of the proposed approach.

4.2 Design scope

SACM provides a specification that defines a metamodel for representing structured assurance cases. It is composed of the following components:

The development of the SACM argumentation notation in this thesis adopted the iterative notation design process, as previously discussed in Chapter 3. The first iteration of development and evaluation is the focus of this chapter, where the elements that we considered important and necessary in representing assurance cases using SACM will be visualised. An explanation is provided regarding the identification of the elements of the SACM argumentation to create the design path, and the notation development is described further.

4.3 User involvement

We considered the following types of notation users to be involved in the development and evaluation of the proposed notation:

- OMG - SACM committee

This type of user is a group of individuals who are involved in the development of the SACM specification. This type of user provides valuable input in the development of the proposed notation since they are considered to be experts in the underlying metamodel and the use of assurance cases.

- Experienced user

This type of user is a group of individuals who have prior experience and knowledge in assurance case development or review using a particular notation. This user is expected to provide valuable feedback regarding the proposed notation from the point of view of someone who regularly uses a particular notation to develop or review assurance cases.

- Novice user

This type of user is a group of individuals who have no prior experience or knowledge related to assurance cases. They are expected to provide objective and unbiased feedback related to the proposed notation.

Figure 4.2 summarises the involvement of each group of notation users in the development and evaluation of the proposed notation in this thesis.

The OMG - SACM Committee are involved in the development phase specifically in the discussion of the design scope and the development of the proposed SACM argumentation notation, such as providing feedback related to the avoidance of clashes with existing notations.

In the evaluation phase, the assurance case expert users are involved together with the novice users. The assurance case expert is expected to provide feedback based on their prior experience and knowledge related to the use of other assurance case notations. The novice user is expected to provide

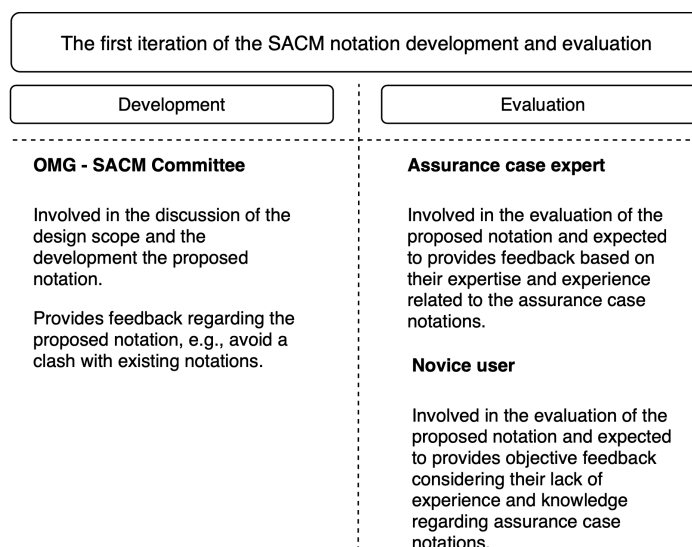


FIGURE 4.2: Notation user involvement in the development and evaluation of the SACM notation

an objective feedback regarding the proposed notation that is not influenced by knowledge of other assurance case notations.

The way in which the users are involved is based on understanding from the literature relating to empirical notation evaluation and also considering the limited time availability of the users to be involved in both the development and evaluation phase. For example, from the literature, we learned that involving the naive users in the development phase can be more complicated rather than involving them in the evaluation phase [6, 16, 51, 53, 54].

Furthermore, the targeted users (with a different experience and knowledge), such as assurance case developers, specialists, engineers, reviewers, and project managers, are expected to use the developed SACM notation.

4.4 Design path

4.4.1 Elements identification

The objective of this sub-step is to identify which elements of the SACM argumentation metamodel will be represented using a visual representation. The SACM argumentation metamodel is the input to complete this sub-step.

Figure 4.3 illustrates the SACM argumentation metamodel from the SACM specification version 2.0. The elements of the SACM metamodel were extracted and presented in Table 4.1.

In SACM, structured arguments are represented explicitly by the ‘Claims’, citation of artifacts via ‘ArtifactReferences’ (e.g. Evidence and Context for claims), and the relationships between these elements [9]. The following relationships can be used to associate between elements:

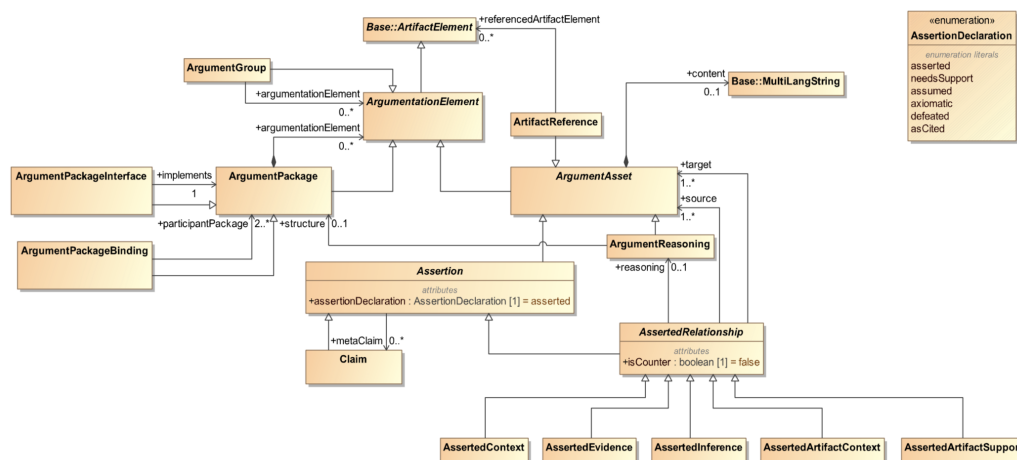


FIGURE 4.3: SACM Argumentation Metamodel from the SACM specification version 2.0 [9]

TABLE 4.1: Extracted SACM argumentation metamodel elements

Element	Type
'ArgumentationElement'	(abstract) class
'ArgumentGroup'	(concrete) class
'argumentationElement'	association relationship
'ArgumentPackage'	(concrete) class
'argumentationElement'	composition relationship
'ArgumentPackageInterface'	(concrete) class
'implement'	association relationship
'ArgumentPackageBinding'	(concrete) class
'participantPackage'	association relationship
'ArgumentAsset'	(abstract) class
'content'	composition relationship
'Assertion'	(abstract) class
'assertionDeclaration', with enumeration literals: - 'asserted' - 'needsSupport' - 'assumed' - 'axiomatic' - 'defeated' - 'asCted'	attribute of 'Assertion' class
'Claim'	(concrete) class
'metaClaim'	association relationship
'AssertedRelationship'	(abstract) class
'isCounter'	attribute of 'AssertedRelationship' class
'reasoning'	association relationship
'source'	association relationship
'target'	association relationship
'AssertedContext'	(concrete) class
'AssertedEvidence'	(concrete) class
'AssertedInference'	(concrete) class
'AssertedArtifactContext'	(concrete) class
'AssertedArtifactSupport'	(concrete) class
'ArtifactReference'	(concrete) class
'referenceArtifactElement'	association relationship
'ArgumentReasoning'	(concrete) class
'structure'	association relationship

- 'AssertedInference' for relationship between 'Claims'
- 'AssertedEvidence' for 'ArtifactReference' (as Evidence) and 'Claims'

- ‘AssertedContext’ for ‘ArtifactReference’ (as Context) and ‘Claims’
- ‘AssertedArtifactSupport’ for ‘ArtifactReference’ (as Evidence) supporting ‘ArtifactReference’ (as Evidence and Context)
- ‘AssertedArtifactContext’ for ‘ArtifactReference’ (as Context) supporting ‘ArtifactReference’ (as Evidence and Context)

The ‘Claim’ and the Relationship in SACM are defined as an ‘Assertion’ where each ‘Assertion’ can be declared to have the following state:

- ‘asserted’ - that indicates an ‘Assertion’ is asserted (default state)
- ‘assumed’ - indicating that the ‘Assertion’ being made is declared by the author as being assumed to be true rather than being supported by further argumentation
- ‘axiomatic’ - indicates that the ‘Assertion’ being made by the author is axiomatically true, so that no further argumentation is needed
- ‘defeated’ - indicating that the ‘Assertion’ is defeated by counter-evidence and/or argumentation
- ‘asCited’ - indicating that because the ‘Assertion’ is cited, the ‘AssertionDeclaration’ should be transitively derived from the value of the ‘AssertionDeclaration’ of the cited ‘Assertion’
- ‘needsSupport’ - indicating that further argumentation has yet to be provided to support the ‘Assertion’

In addition to these core elements for the representation of assurance cases, it is also possible and considered necessary to provide:

- Description of the ‘ArgumentReasoning’ that can be used to provide additional description or explanation of the asserted relationship
- Counter-Argument and Counter-Evidence (via isCounter: Boolean) to support dialectical assurance case constructions
- Modular structured argument (via ‘ArgumentPackage’) including the mechanism to organise a specific selection of the ‘ArgumentElements’ contained within the package (via ‘ArgumentPackageInterface’), and a mechanism to bound two or more ‘ArgumentPackages’ (via ‘ArgumentPackageBinding’)
- An association of a number of ‘ArgumentElements’ to a common group (via ‘ArgumentGroup’)

- A references ‘Claims’ concerning (i.e., about) the ‘Assertion’ via ‘MetaClaim’ relationship, for example, to be used to represents a claim regarding the confidence in the ‘Assertion’

To get comprehensive information regarding the representation of assurance case using SACM, it is important to study the SACM metamodel as a complete specification. The information is gathered not only from the Argumentation part of the metamodel, but also from other related components, specifically, from Assurance Case Base metamodel where the root (base) class of the SACM metamodel is located.

The root class of the SACM metamodel is the ‘SACMElement’ class. It is an abstract class that has three attributes: ‘gid’, ‘isCitation’, and ‘isAbstract’. We considered these attributes are important for the representation of assurance cases using SACM since they provide:

- a unique identifier that is unique within the scope of the model instance (via ‘gid’ attribute) to indicate the element ID.
- an Abstract definition to a particular element (via ‘isAbstract’ attribute) that can be used to indicate whether a particular element is considered to be used as part of a pattern or template (to support the assurance case pattern construction).
- a Citation definition to indicate whether the SACMElement cites another SACMElement (via ‘isCitation’ attribute).

These attributes are inherited from the SACMElement class in the Assurance Case Base metamodel to the root class of the SACM argumentation metamodel (i.e. ‘ArgumentationElement’ class) and other classes in the SACM argumentation metamodel via generalisation relationship so the attributes should be represented as part of the SACM argumentation notation.

After studying the SACM metamodel (including identifying the concrete classes) and considering the semantic importance of the elements for the representation of assurance cases using SACM, the identified SACM elements that we considered necessary to be visualised using a visual representation and included in the first iteration of the development of the SACM argumentation notation are shown in Table 4.2.

The following elements are not represented using visual representations since they are defined as an abstract class in SACM argumentation metamodel, however, the information contained within these elements can be useful to help creating the visual representation of other elements such as information regarding the ‘assertionDeclaration’ attribute that can be used to declare specific state of an assertion (e.g. ‘Claim’):

1. ‘ArgumentationElement’

TABLE 4.2: Identified elements to be visualised as part of the SACM notation design (First iteration)

Element	Type
'gid'	class attribute - inherited from root class of the SACM metamodel
'isCitation'	class attribute - inherited from root class of the SACM metamodel
'isAbstract'	class attribute - inherited from root class of the SACM metamodel
'Claim'	(concrete) Class
'asserted'	enumeration literals of the 'assertionDeclaration' attribute
'needsSupport'	enumeration literals of the 'assertionDeclaration' attribute
'assumed'	enumeration literals of the 'assertionDeclaration' attribute
'axiomatic'	enumeration literals of the 'assertionDeclaration' attribute
'defeated'	enumeration literals of the 'assertionDeclaration' attribute
'asCited'	enumeration literals of the 'assertionDeclaration' attribute
'ArtifactReference'	(concrete) Class
'isCounter'	attribute of 'AssertedRelationship' class
'AssertedInference'	(concrete) Class
'AssertedEvidence'	(concrete) Class
'AssertedContext'	(concrete) Class
'AssertedArtifactSupport'	(concrete) Class
'AssertedArtifactContext'	(concrete) Class
'ArgumentReasoning'	(concrete) Class
'ArgumentPackage'	(concrete) Class
'ArgumentPackageInterface'	(concrete) Class
'ArgumentPackageBinding'	(concrete) Class
'ArgumentGroup'	(concrete) Class
'metaClaim'	association relationship
'implements'	association relationship
'participantPackage'	association relationship

2. 'ArgumentAsset'
3. 'Assertion'
4. 'AssertedRelationship'

We considered the following elements are not necessarily to be visualised using a visual representation since they can be used to define the compositional rules of the elements when combined in a diagram. This decision is made also in considering the complexity of the visual notation:

1. 'target' association relationship: can be used to define the target (ending point) of the relationship.
2. 'source' association relationship: can be used to define the source (starting point) of the relationship.
3. 'reasoning' association relationship: can be used to define that the reasoning of a particular relationship can be described by an 'ArgumentReasoning' (i.e. an 'ArgumentReasoning' can be associated to a particular relationship to provides further description of the reasoning involved).
4. 'structure' can be used to define an optional reference to another 'ArgumentPackage' that provides the detailed structure of the argument being described by the 'ArgumentReasoning'.
5. 'argumentationElement' association can be used as the rule for the application of the 'ArgumentGroup' visual representation (i.e. argumentation elements can be grouped together using

the ‘ArgumentGroup’ visual representation)

6. ‘argumentationElement’ composition relationship can be used to define that ‘ArgumentPackage’ contain a collection of ‘ArgumentationElements’ forming a structured argument.
7. ‘referenceArtifactElement’ association relationship can be used to define a reference to a collection of ‘ArtifactElements’.
8. content composition relationship can be used to define that the content of the ‘ArgumentAsset’ can be defined in possibly multiple languages.

After the elements that we considered to be visualised using visual representations are identified, these elements need to be prioritised in the next sub-step.

4.4.2 Element prioritisation

The elements identified in the previous sub-step need to be prioritised to define the order of the notation design. This is important since we need to create a unique visual representation of an element by utilising and combining visual variable values and in order to avoid overload in the visual representation, we cannot use the same combination of variables to represent different elements (i.e. representing more than one elements using an identical visual representation). It is important that priority in the use of visual variables is given to the most important elements. In the case of SACM argumentation elements, we prioritised the core elements of the metamodel first and then the supporting elements.

In SACM, structured arguments are represented explicitly by the ‘Claims’, citation of artifacts such as Evidence and Context for claims (via ‘ArtifactReferences’), and the relationships between these elements. The core of any argument is a series of claims (premises) that are asserted to provide sufficient reasoning to support a (higher-level) claim (a conclusion). Therefore, we prioritised the ‘Claim’ class, ‘ArtifactReference’, and the ‘AssertedRelationship’ class as they represent the core elements and the fundamental concepts for the representation of the structured arguments. Following the core elements, we also prioritised the inherited attributes that can be used to support the core elements such as the attributes that are inherited from the root class: ‘gid’, ‘isCitation’, and ‘isAbstract’, and the attributes from ‘Asserted’ class (i.e. ‘assertionDeclaration’) that can be used to define a specific state of a ‘Claim’ and ‘AssertedRelationship’.

Following the core elements and the attributes that can be used to declare a specific state of the core elements, the supporting elements such as ‘ArgumentReasoning’, ‘MetaClaim’, ‘ArgumentPackages’, and ‘ArgumentGroup’ were added to the prioritisation list. The ordering process of the supporting elements is based on the importance of their semantics in supporting the core elements.

To summarise, Table 4.3 presents the elements that are listed based on their prioritised order.

TABLE 4.3: Prioritised elements

Element	Type
'Claim'	(concrete) class
'gid'	class attribute - inherited from root class of the SACM metamodel
'asserted'	enumeration literals of the 'assertionDeclaration' attribute
'needsSupport'	enumeration literals of the 'assertionDeclaration' attribute
'assumed'	enumeration literals of the 'assertionDeclaration' attribute
'axiomatic'	enumeration literals of the 'assertionDeclaration' attribute
'defeated'	enumeration literals of the 'assertionDeclaration' attribute
'asCited'	enumeration literals of the 'assertionDeclaration' attribute
'isCitation'	class attribute - inherited from root class of the SACM metamodel
'isAbstract'	class attribute - inherited from root class of the SACM metamodel
'AssertedInference'	(concrete) Class
'isCounter'	attribute of 'AssertedRelationship' class
'ArtifactReference'	(concrete) Class
'AssertedEvidence'	(concrete) Class
'AssertedContext'	(concrete) Class
'AssertedArtifactSupport'	(concrete) Class
'AssertedArtifactContext'	(concrete) Class
'ArgumentReasoning'	(concrete) Class
'metaClaim'	association relationship
'ArgumentPackage'	(concrete) Class
'ArgumentPackageInterface'	(concrete) Class
'implements'	association relationship
'ArgumentPackageBinding'	(concrete) Class
'participantPackage'	association relationship
'ArgumentGroup'	(concrete) Class

After the elements have been prioritised, the next step is the creation of the design path that is described in the next sub-section. It is important to be noted; different notation designer might have a different opinion in determining the order of the elements. In this thesis, we used the above order as an example to determine the order of the elements.

4.4.3 Design path creation

The design path is created in order to show the order of the design of the identified elements and to show the other elements that are involved in the design process. The input for the creation of the design path is the list of the prioritised elements together with the other elements that are involved in the creation of the visual representations of the elements (e.g. the root class of the SACM metamodel that is inheriting its attributes to the elements of the SACM metamodel).

The design path of the SACM argumentation notation starts from root class of the SACM metamodel (i.e. 'SACMElement' class) that is located in the Assurance Case Base metamodel area. As mentioned in the previous sub-section, the 'SACMElement' class is an abstract class that has three attributes (i.e. 'gid', 'isCitation', and 'isAbstract') that can be inherited to its sub-classes and are considered important to be visualised.

The root class of the SACM metamodel is connected through generalisation relationship to the root class of the SACM argumentation metamodel (i.e. 'ArgumentationElement' class) via 'ModelElement' class (sub-class of the 'SACMElement' class) and 'ArtifactElement' class (sub-class of the

‘ModelElement’ class), in this case, the ‘ArtifactElement’ class are the super-class of the ‘ArgumentationElement’ class. Both the ‘ModelElement’ class and the ‘ArtifactElement’ class are not part of the SACM argumentation metamodel and defined as an abstract class with no attributes that can be inherited to their sub-classes.

The ‘ArgumentationElement’ class is an abstract class without any attribute that can be inherited. From this class, the design path continues to the ‘ArgumentAsset’ class since the sub-classes of the ‘ArgumentAsset’ are the classes that contain the core concept to represent assurance cases using SACM (e.g. ‘Claim’ class, ‘ArtifactReference’ class, and the ‘AssertedRelationship’ class).

The ‘ArgumentAsset’ is an abstract class without any attributes that can be inherited. From the ‘ArgumentAsset’ class, we then continue to the ‘Assertion’ class since the ‘Claim’ class, which is the first element to be designed, is a sub-classes of ‘Assertion’. The ‘Assertion’ class is an abstract class with an attribute: ‘assertionDeclaration’. This attribute has six enumeration literals that can be used to declare specific state of an ‘Assertion’ and is inherited to the sub-classes of the ‘Assertion’ class (i.e. ‘Claim’ class and ‘AssertedRelationship’ class).

From the ‘Assertion’ class, the design path, essentially continues by following the order of the prioritised elements that was defined in the previous section. However, there are several classes that need to be inserted in the middle of the path such as the ‘AssertedRelationship’ class (as an abstract super-class of the ‘AssertedInference’) before the ‘AssertedInference’ class. The insertion of the ‘AssertedRelationship’ class to the path is important, specifically, related to creation of the visual representation of its sub-classes, such as ‘AssertedInference’, ‘AssertedEvidence’, and ‘AssertedContext’, since there are design constraints that can be gathered from the ‘AssertedRelationship’ class (e.g. from the semantics) that must be inherited to its sub-classes. In such cases, the abstract class can help define the design constraints that can be applied to its concrete sub-classes.

To summarise, the design path is created as illustrated in Figure 4.4 for the development of the SACM argumentation notation. We modified the SACM argumentation metamodel by adding two classes, the ‘SACMElement’ class (root class of SACM metamodel) and the ‘ModelElement’ class, to show the complete created design path of the SACM argumentation notation that starts from the root class of the SACM metamodel and continues to the elements of the SACM argumentation metamodel.

The numbers in Figure 4.4 indicate the order of the design path. The design path starts from the ‘SACMElement’ class where several attributes from this class can be inherited to the elements in the SACM argumentation metamodel, and then continues to the next number until the last number, 32, where the design path stops in the ‘ArgumentGroup’ class. If the element in the design path is part of the identified elements that are considered necessary to have a visual representation, then the visual representation of that element is created by utilising visual variables and considering the defined and inherited design constraints. The tabular view of the created design path also can be seen in Table 4.4.

In the next section we discuss the design constraints that are considered in the development of the SACM notation.

4.5 Design constraints

In this section, the considered design constraints for the creation of the SACM argumentation notation are described, including the selection of visual variables, existing notations, existing notation design principles, and the notation usability factors.

4.5.1 Selected visual variables

For the development of SACM argumentation notation, the following visual variables are considered: Shape, Position/Location, Brightness, Size, and Texture.

4.5.2 Considered existing notations

We considered the following existing notations in the development of the SACM argumentation notation:

- Goal Structuring Notation (GSN)

GSN is considered since it is one of the existing and established notations in the assurance case domain that is widely adopted both in industry and research. The development of the SACM argumentation notation will consider GSN in particular as an index visual representation in order to provide a clue to notation users who are familiar with GSN (e.g. in using existing conventions to represent Claim)

- Claim-Argument-Evidence (CAE)

CAE is considered since, like GSN, it is one of the existing and established assurance case notations. In the development of SACM argumentation notation, CAE is specifically considered in the context of avoiding a visual representation clash for users familiar with CAE notation.

- Unified Modeling Language (UML)

UML is one of the existing and established notations that is published by the Object Management Group (OMG). Since SACM is also published by the OMG, the visual representation of the SACM must take account of the UML notation to avoid visual representation clashes between UML and SACM. For example, a discussion with the OMG committee was held related to the proposed visual representation of the 'ArtifactReference' as a document (as illustrated in Figure 4.5 (a)). This design needed to be modified since it was considered too similar to the

Perceptual discriminability principle was considered in the development of the SACM argumentation notation in order to create a unique visual representation so it can be distinguished from other visual representations.

Since the number of the elements of SACM that need to be visualised is high (25 elements), and considering the *graphic complexity* of the notation (i.e. managing the number of the visual representations of a notation [3]), we also used the source and target elements in the diagram to visually distinguished elements. In other words, it is possible that, for example, two visual representations look identical in isolation, however, they can still be distinguished by their position and by identifying the source and target elements in a diagram.

- Redundant coding

Redundant coding was adopted in the development of SACM argumentation notation as an effort to increase the perceptual discriminability between elements by using a combination of multiple visual variables and their unique value in creating visual representations.

4.5.4 Considered notation usability

We considered the following usability factors in the development of the SACM argumentation notation:

- Ease of drawing

Besides adopting the semantic transparency principle in order to create an intuitive visual representation, the ease of drawing notation usability factor is also considered. This facilitates users of the notation who wish to create a model by for example hand-sketching on a whiteboard or paper (e.g. during lectures in a class room). This means the created visual representations of the SACM argumentation are created as simple as possible by not utilising and combining an excessive number of visual variables. This is one of the reasons why not all visual variables are adopted in the development of the SACM notation; two visual variables that are not being used are colour and orientation.

- The use of colour

In the development of the SACM argumentation notation, the visual variable Colour was not adopted. This is because it is sensitive to variations in visual perception such as color blindness and screen or printer characteristics e.g. the use of black-and-white printers.

- Tool support implementation

The tool support implementation was also considered as the SACM visual representation may often be implemented in a software tool. This is a common use case for many users.

4.6 SACM argumentation notation

The development of the SACM argumentation notation is described in this section. The development considered the created design path (as the output of the Step 1) and the design constraints (as the output of Step 2). The description of the development process starts from the root class of the SACM metamodel where we identified related information that can be used to create the SACM argumentation notation.

4.6.1 ‘SACMElement’

‘SACMElement’ is the base or root class of the SACM metamodel. It is located in the Structured Assurance Base classes therefore it is not part of the SACM argumentation metamodel. However, since ‘SACMElement’ is a parent class for all elements of the SACM argumentation metamodel, we need to identify relevant information such as class attributes that can be inherited to the elements for the purpose of the development of the visual representations.

‘SACMElement’ is an abstract class. Its semantics are as follows:

"All the elements of a structured assurance case effort created with SACM correspond to a ‘SACMElement’".

Since the ‘SACMElement’ is an abstract class and is not part of the SACM argumentation metamodel, its visual representation is not created. It is still necessary to consider whether the element imposes any constraints on the visual representation of other elements.

The ‘SACMElement’ has three class attributes:

- ‘gid’:String[0..1] – a unique identifier that is unique within the scope of the model instance
- ‘isCitation’[1]=false – a flag to indicate whether the ‘SACMElement’ cites another ‘SACMElement’
- ‘isAbstract’[1]=false – a flag to indicate whether the ‘SACMElement’ is considered to be abstract. For example, this can be used to indicate whether an element is part of a pattern or template.

The attributes of the ‘SACMElement’ are applied to the elements of the SACM argumentation via the inheritance process. These attributes are inherited to the elements of the SACM argumentation metamodel via ‘SACMElement’ sub-classes (‘ModelElement’ class and the ‘ArtifactElement’ class) for the purpose of the development of the visual representations.

As follows we introduced the visual representation design constraints of the ‘SACMElement’ attributes that can be used consistently by the elements of the Argumentation metamodel:

- ‘gid’

The ‘gid’ attribute is represented using text that is unique for each SACM element that can be written using a regular text type or in bold (when considered it is necessary by the user to put emphasise for the element ID). The use of text is based on the attribute type of the ‘gid’ which is String.

The application of the ‘gid’ attribute is optional by any instantiation of the SACM elements.

Since ‘gid’ is represented using text, the only visual variable that is used related to representation of ‘gid’ is the Location/Position visual variable that is defined by the position of the element ID when applied to a particular visual representation.

When considered necessary by the notation user, the element ID can be applied and located with the following rules to the visual representation of a particular element:

- For 1D-type visual representation, such as line to represent a relationship, an element ID can be located near the middle of the line. However, such application is not common in the visual modelling language areas. Therefore, an element ID is not recommended to be applied for 1D-type visual representation.
- For 2D-type visual representation, such as a Rectangle, an element ID can be located within (top-left corner) of the visual representation.

The compositional rule of the ‘gid’ attribute can be inherited as a design constraint that can be applied consistently to other elements of SACM argumentation metamodel.

- ‘isCitation’

Based on the semantics, ‘isCitation’ can be applied to any sub-classes of the ‘SACMElement’, including the elements of the SACM argumentation metamodel, in order to indicate whether the ‘SACMElement’ cites another ‘SACMElement’.

The creation of the ‘isCitation’ visual representation must be suitable when being applied to the visual representation of the instances of the ‘SACMElement’, including the elements of the SACM argumentation metamodel (e.g. ‘Claim’, ‘AssertedInference’ and ‘ArtifactReference’).

Considering the semantics of the ‘isCitation’, the visual representation is created as square brackets. This visual representation is inspired by a common approach for a citation in a scientific writing style. Figure 4.7 illustrates the visual representation of the ‘isCitation’. The location of the cited element in the diagram can be presented at the top-left corner of the visual representation (e.g. in a specific argument package location).



FIGURE 4.7: Citation visual representation

The visual variables that are used to create the ‘isCitation’ are similar to the visual variables that are used to create ‘asCited’ that can be described as follow:

- Shape: "Lines" that form square brackets symbol ("[]").
- Location/Position: The cited element must be located between the square brackets. When used in a diagram the placement of the citing element follows the placement rules of the cited element.
- Size: "1 pt" line size thickness for the lines that form the square brackets.
- Texture: "Solid" for the texture of the line.

The visual representation of the ‘isCitation’ can be used by the elements of the Argumentation metamodel when applied to the inherited ‘isCitation’ attribute as a design constraint.

- ‘isAbstract’

Considering the semantics, the visual representation of the ‘isAbstract’ must be created to be applicable when being implemented by the visual representations of the SACM elements including the SACM argumentation metamodel elements.

The visual representation of ‘isAbstract’ is illustrated in Figure 4.8. It is visually represented using a dashed line that is inspired by the UML Template visual representation since their purpose is similar which is to indicate whether an element is declared as an abstract or can be used as part of a template. A visual representation clash with existing notation, in this case with UML, is avoided.



FIGURE 4.8: The use of dashed line to denote ‘isAbstract’

The visual variable is used as a design constraint to create an abstract element is the Texture visual variable with value "Dash line". This design constraint can be inherited and applied consistently by the elements of the SACM argumentation when being declared as an abstract element.

The application of the ‘isAbstract’ visual representation depends on the visual representation of the element that is declared as an abstract element. For example, in the case of a ‘Claim’ that

is declared as being abstract, this ‘Claim’ is called an ‘abstract Claim’. An ‘abstract Claim’ is a ‘Claim’ that is declared as part of a pattern or template.

‘SACMElement’ has two association relationships:

- ‘citedElement’:‘SACMElement’[0..1] - that can be used as a reference to another ‘SACMElement’ that the ‘SACMElement’ cites. The source and target element of this association is the ‘SACMElement’ class.
- ‘abstractForm’:‘SACMElement’[0..1] - that can be used as an optional reference to another abstract ‘SACMElement’ to which this concrete ‘SACMElement’ conforms. The source and target element of this association is the ‘SACMElement’ class.

The association relationships of the ‘SACMElement’ provides information regarding the association between the instantiation of the ‘SACMElement’ that can be applied including by the elements of the SACM argumentation metamodel.

4.6.2 ‘ModelElement’

‘ModelElement’ is the sub-class of the ‘SACMElement’ that is connected to the elements of the SACM argumentation metamodel through generalisation relationship via its sub-class (‘ArtifactElement’).

‘ModelElement’ is located in the same area as ‘SACMElement’ (i.e. the Structured Assurance Base classes). Since ‘ModelElement’ is abstract and is not part of the SACM argumentation metamodel, its visual representation is not created. However it is necessary to consider any constraints it may impose on concrete classes due to inheritance in the metamodel.

‘ModelElement’ class has no attributes other than the inherited attributes from its super-class (i.e. ‘SACMElement’ class). These inherited attributes (i.e. ‘gid’, ‘isAbstract’, and ‘isCitation’) can be inherited to its sub-class, ‘ArtifactElement’ class, via generalisation relationship.

‘ModelElement’ has five composition relationships:

- ‘name’:LangString[1] (composition) – the name of the ‘ModelElement’.
- ‘implementationConstraint’: ‘ImplementationConstraint’ [0..*] (composition) – a collection of implementation constraints.
- ‘description’: ‘Description’[0..1] (composition) – the description of the ‘ModelElement’.
- ‘note’:‘Note’[0..*] (composition) – a collection of notes for the ‘ModelElement’.

- ‘taggedValue’: ‘TaggedValue’ [0..*] (composition) – a collection of ‘TaggedValues’. ‘Tagged-Values’ can be used to describe additional features of a ‘ModelElement’.

In the context of SACM argumentation notation, the ‘name’ composition relationships is used to represent the name of the element where in this case it is represented via ‘gid’ attribute to indicate the element unique ID. The ‘description’ composition relationships is used to represent the description that is required for a particular element, e.g., the description of the ‘Claim’ element. The other composition relationships (i.e. ‘implementationConstraint’, ‘note’, and ‘taggedValue’) are considered not urgently required to be represented and included in first iteration design of the SACM argumentation notation.

The following classes are listed in the design path as the next class to be processed after the ‘ModelElement’ class. We have identified that these classes are defined as an abstract class (hence do not require any visual representation) and have no attributes in the SACM specification document:

- ‘ArtifactElement’
- ‘ArgumentationElement’
- ‘ArgumentAsset’

Following the ‘ArgumentAsset’ the next class to be processed based on the design path is the ‘Assertion’ class.

4.6.3 ‘Assertion’

‘Assertion’ is an abstract class that is defined as:

"‘Assertions’ are used to record the propositions of Argumentation (including both the Claims about the subject of the argument and the structure of the Argumentation being asserted). Propositions can be true or false, but cannot be true and false simultaneously. Structured arguments are declared by stating claims, citing evidence and contextual information, and asserting how these elements relate to each other."

Since the ‘Assertion’ is an abstract class, therefore, its visual representation is not created.

Other than the inherited attributes from ‘SACMElement’ class (‘gid’, ‘isAbstract’, and ‘isCitation’), ‘Assertion’ class has an attribute named as the ‘assertionDeclaration’ with the following enumeration literals:

- ‘asserted’ - the default enumeration literal, indicating that an ‘Assertion’ is asserted.
- ‘needsSupport’ - a flag indicating that further argumentation has yet to be provided to support the ‘Assertion’.

- ‘assumed’ - a flag indicating that the ‘Assertion’ being made is declared by the author as being assumed to be true rather than being supported by further argumentation.
- ‘axiomatic’ - a flag indicating that the ‘Assertion’ being made by the author is axiomatically true, so that no further argumentation is needed.
- ‘defeated’ - a flag indicating that the ‘Assertion’ is defeated by counter-evidence and/or argumentation.
- ‘asCited’ - a flag indicating that because the ‘Assertion’ is cited, the ‘AssertionDeclaration’ should be transitively derived from the value of the ‘AssertionDeclaration’ of the cited Assertion.

The ‘assertionDeclaration’ attribute of the ‘Assertion’ class can be inherited to its sub-classes that can be used to declare a specific state (using a particular enumeration literal) of the instantiation of the classes. Therefore, the enumeration literals of the ‘assertionDeclaration’ can be inherited as design constraints to the sub-classes of the ‘Assertion’ class and should be applied consistently for the purpose of the creation of visual representation to show family resemblance among related visual representations.

We introduced the visual representation design constraints of the enumeration literals of the ‘assertionDeclaration’ attribute that must be respected and applied consistently for the ‘Assertion’ class as follows. The created visual representation design constraints considered the applicability when implemented to each sub-class of the ‘Assertion’ class:

- **‘asserted’**

This is the default enumeration literal; therefore, the visual representation of the ‘asserted’ element is the visual representation of the element itself.

- **‘needsSupport’**

Based on the semantics of ‘needsSupport’, the visual representation of the ‘needsSupport’ is created as three dots that can be applied consistently by the (concrete) sub-classes of the ‘Assertion’ class for the creation of the visual representation. The tree dots visualisation is inspired from a convention used in the text for "yet to be supported" (...).

The following visual variables are used to create the visual representation of the ‘needsSupport’ that can be used as design constraints that can be applied by the sub-classes of the ‘Assertion’ class:

- Shape: "3 Circles".

- Brightness: "Dark".
- Location:
 - * "Bottom part of the rectangle" when being applied in a 'Claim'.
 - * "on the middle of the line" when being implemented in a 'AssertedRelationship' type.
- Size: "1pt": as the size of the line that form the Circle; "0.25" cm for the width and height of the circles (adjustable relative to the application in a diagram).

The use of three dots is expected to indicate to the notation users that this type of element is not completed and further argumentation element has yet to be provided to support this element.

- **'assumed'**

Based on the semantics, the 'assumed' visual representation is created as a line gap (to indicates an assumption) that can be applied consistently by the (concrete) sub-classes of the 'Assertion' class for the creation of the visual representation. The gap is supposed to indicate that the notation user should not connect any support to the element, i.e. there is nothing to attach to support it. This is because assumptions are, by definition, is unsupported.

The following visual variables are used to create the visual representation of the 'assumed' that can be used as design constraints that can be applied by the sub-classes of the 'Assertion' class:

- Shape: "A line with a gap in the middle of the line (with two vertical lines at the line-end where the line gap is):.
- Location:
 - * "Bottom part of the rectangle" when being applied in a 'Claim'
 - * "On the middle of the line" when being implemented in a 'AssertedRelationship' type
- Size: "1pt" as the size of the line.
- Texture: "Solid" for the texture of the line.

- **'axiomatic'**

Considering the semantics of the 'axiomatic', a thick line (3 points line thick size) is used to visually represents the 'axiomatic' to denotes that no further argumentation is needed to support the 'Assertion' that is declared as 'axiomatic'. A thick line is adopted to indicate a termination, i.e. nothing further is required to support the element.

The following visual variables are used to create the visual representation of the ‘axiomatic’ that can be used as design constraints that can be applied by the sub-classes of the ‘Assertion’ class:

- Shape: "A line".
- Location/Position:
 - * "Below" the rectangle when applied to a Claim.
 - * "In the middle of the line" when being applied to an ‘AssertedRelationship’ type, and positioned vertically relative to the line if the line of the ‘AssertedRelationship’ type drawn horizontally, and positioned horizontally if the line of the ‘AssertedRelationship’ type drawn vertically
- Size: "3pt" as the size of the line; for the length of the line is relatively following the size of the rectangle that is placed above the line (when being applied in a ‘Claim’). When being applied in an ‘AssertedRelationship’ type, the length of the line is relatively smaller than the length of the relationship line.
- Texture: "Solid" for the texture of the line.

- **‘defeated’**

By considering the semantics, the ‘defeated’ is represented using a cross that can be placed on top of the visual representation declared as being defeated. The cross is adopted to indicate a crossing-out, i.e., the element has been defeated by counter-evidence/argument.

The visual representation of the ‘defeated’ is created by utilising and combining the following visual variables:

- Shape: "A-double-lines" that form a "cross" or "X" letter/symbol.
- Location: The cross symbol is placed on top of the rectangle (when applied to a ‘Claim’); and in the middle of the line, when applied in an ‘AssertedRelationship’ type such as ‘AssertedInference’.
- Size: "1 pt" line size thickness for the lines that form a "cross" or "X" letter/symbol. When applied to a ‘Claim’, the cross must be drawn on top of the ‘Claim’ so the ‘Claim’ is covered by the cross symbol (the size of the cross (not the line thickness) is adjusted relatively to the size of the rectangle); and when applied to an ‘AssertedRelationship’ type, the cross must be drawn relatively in the size (of the cross, not the line thickness) of the line-head of the relationship (further explanation and example can be found, for example,

in the section of the ‘AssertedInference’).

- Texture: "Solid" for the texture of the line.

- **‘asCited’**

By considering the semantics, the ‘asCited’ visual representation must show that the ‘Assertion’ that is declared as an ‘asCited’ ‘Assertion’ is cited another ‘Assertion’. The ‘asCited’ is visually represented as a citation symbol that is commonly used in scientific writing field that is denoted using square brackets where the cited element is located between the square brackets.

The following visual variables are used to create the visual representation of the ‘asCited’ that can be inherited and applied consistently by the sub-classes of the ‘Assertion’ class:

- Shape: "Lines" that form square brackets symbol ("[]").
- Location/Position: The citing element must be located between the square brackets. When used in a diagram the placement of the citing element follows the placement rules of the cited element.
- Size: "1 pt" line size thickness for the lines that form the square brackets.
- Texture: "Solid" for the texture of the line.

The ‘Assertion’ class has two sub-classes: ‘Claim’ and ‘AssertedRelationship’. The specification of each enumeration literal of the ‘assertionDeclaration’ attribute, in terms of for the application to develop a visual representation for each concrete sub-classes of the ‘Assertion’ class, is described further in the ‘Claim’ sub-section and sub-sections of the sub-classes of the ‘AssertedRelationship’ class.

‘Assertion’ has an association relationship named ‘metaClaim’. It can be used as references ‘Claims’ concerning (i.e., about) the ‘Assertion’ (e.g., regarding the confidence in the ‘Assertion’). Further explanation regarding this attribute is discussed in sub-section ‘metaClaim’ in this chapter.

4.6.4 ‘Claim’ and its variants

‘Claim’ is a concrete class. Its semantics is described as follow according to the SACM specification:

"‘Claims’ are used to record the propositions of any structured argument contained in an ‘Argument-Package’. Propositions are instances of statements that could be true or false, but cannot be true and false simultaneously. The core of any argument is a series of claims (premises) that are asserted to provide sufficient reasoning to support a (higher-level) claim (a conclusion)."

Since ‘Claim’ is a concrete class, therefore, its visual representation needs to be created. The visual representation of a ‘Claim’ is rendered as a rectangle where the claim statement can be written within the rectangle. Figure 4.9 shows the visual representation of a ‘Claim’.

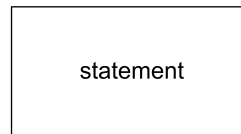


FIGURE 4.9: ‘Claim’ visual representation

In the creation of the ‘Claim’ visual representation, the icon-type visual representation is considered first. Due to the difficulty in finding the icon-type visual representation that can represent its semantic, we considered to represent the visual representation of the ‘Claim’ class using the index-type visual representation (adopting visual representation of an existing assurance case notation) to provide a clue to the users who are familiar with the existing notation. In this context, the visual representation of the GSN Goal is adopted to represent the visual representation of a ‘Claim’ since they have similar semantics.

The visual representation of the ‘Claim’ is created based on multiple visual variables such as Shape, Brightness, and Location. Colour is not used in the creation of ‘Claim’. Related to the visual variables used and other design constraints considered in the creation of ‘Claim’ can be read further in Appendix A.1.1.

Regarding the compositional rules of the ‘Claim’, we refer to the specification of the ‘Claim’ class that is described in the SACM standard [9].

A ‘Claim’ is an instantiation of the ‘Assertion’ class. A ‘Claim’ can be associated to any instances of the ‘Assertion’ class, either it is a ‘Claim’ or a specific type of an ‘AssertedRelationship’ such as ‘AssertedInference’ or ‘AssertedEvidence’ relationship.

When used in a diagram, the main ‘Claim’ is located at the top-level of the diagram. It can be supported by other ‘Claims’, that must be placed below the main ‘Claim’, and must be connected via ‘AssertedInference’ relationship. The positioning choices for a ‘Claim’ element is inspired by the GSN approach where the supporting ‘Claims’ are located (vertically) below the top-level ‘Claim’. The positioning of the contextual element in SACM is also inspired by the GSN approach, where the contextual elements are placed horizontally to the target element.

A ‘Claim’ in SACM can also be supported by other ‘Claims’ that is used as a ‘Claim’ that provides a necessary context to the supported ‘Claim, in this case, the supporting ‘Claim’ must be located on the right/left side of the supported ‘Claim’ and must be associated via ‘AssertedContext’ relationship.

A 'Claim' can also be supported by more than one 'ArtifactReferences' that are used as a reference to an evidential information. In this case, the 'ArtifactReferences' must be located below the 'Claim' and must be associated via 'AssertedEvidence' relationship.

In another case, more than one 'ArtifactReferences' can also be used as a reference to an contextual information to scope and provides context to a particular 'Claim'. In this case, the 'ArtifactReferences' must be located horizontally relative to the 'Claim' and must be connected via 'AssertedContext' relationship.

The 'Claim' statement should be written in a single proposition statement in the form of a noun-phrase + verb-phrase sentence. The 'Claim' statement rule is adopted from the GSN Goal statement rule. Figure 4.10 illustrates the example of a Claim with a statement described within the visual representation.

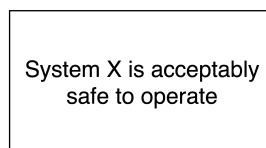


FIGURE 4.10: 'Claim' statement example

The 'Claim' class has inherited attributes: 'gid', 'assertionDeclaration', 'isAbstract', and 'isCitation'.

In SACM, a 'Claim' can be declared into a specific type. The variants of 'Claim' is created based on its inherited attributes (except 'gid' since it is used to define element ID).

The enumeration literals of the 'assertionDeclaration' attribute can be used to declare the 'Claim' into the following specific types:

- 'assumed Claim'. It is visualised as a 'Claim' with a line gap on the bottom part of the 'Claim' to deliver an assumption meaning which indicates that there is no supporting element that can be attached to this element since it is defined as an assumption.
- 'axiomatic Claim'. It is visualised as a 'Claim' with a thick line placed below the 'Claim' to indicate that no further argumentation is needed to support this 'Claim' since it is defined as an axiomatic.
- 'defeated Claim'. It is visualised as 'Claim' with a cross placed on top of the 'Claim' to indicate this 'Claim' is defeated by a counter argument/evidence.
- 'asCited Claim'. It is visualised as a 'Claim' placed within square-brackets to indicate that this 'Claim' is a citation 'Claim' and further explanation about this claim is presented in another

argument structure. The ‘asCited Claim’ notation also can be combined with the others claim, e.g., ‘asCited Claim’ citing an ‘assumed Claim’.

- ‘needsSupport Claim’. It is visualised as a ‘Claim’ with three dots placed at the bottom part of the rectangle to indicate further argumentation has yet to be provided to support this ‘Claim’.

A ‘Claim’ also can be represented as an ‘abstract Claim’ (the application of the inherited ‘isAbstract’ attribute). This represents a ‘Claim’ defined in an abstract form that requires instantiation for a particular argument. Abstract elements, such as an ‘abstract Claim’, are used to support the construction of assurance case patterns. To provide a visual clue to abstract elements they are rendered using dash-lines.

The application of the inherited ‘isCitation’ attribute to the ‘Claim’ element is resulting similar result when applying the ‘asCited’ enumeration literal of the inherited ‘assertionDeclaration’ attribute to the ‘Claim’ element.

Figure 4.11 shows the different types of ‘Claim’ in SACM. For each different type of ‘Claim’, the basic visual representation of a ‘Claim’ (i.e. a rectangle) is used, a necessary decoration is added to present a particular meaning of the assertion declaration (i.e. specific type of an assertion such as a ‘Claim’).

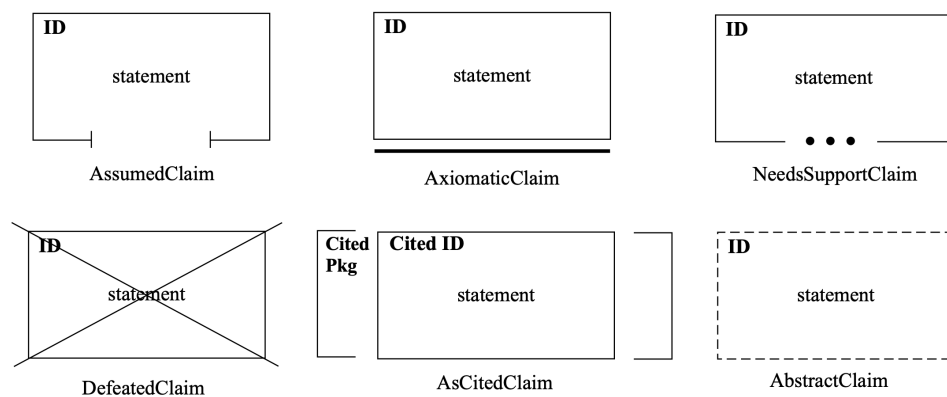


FIGURE 4.11: Different types of ‘Claim’ in SACM

The decoration of each type of ‘Claim’ is used consistently to define a specific type of ‘AssertedRelationship’ in SACM in order to show family resemblance between these related elements (i.e. to show that these elements has semantic similarities).

‘gid’

The application of the ‘gid’ attribute is an optional by any instantiation of the SACM elements, including ‘Claim’ class. Figure 4.12 illustrates the example of the representation of ‘gid’ attributes when applied in a ‘Claim’ visual representation. It is located within (top-left corner) of the visual representation.



FIGURE 4.12: The application of element identifier (ID) in a ‘Claim’ visual representation

‘asserted Claim’

According to the semantics, ‘asserted’ is the default enumeration literals of any ‘Assertion’, hence it is applied to the ‘Claim’ which can indicate that it is the default state of the ‘Claim’. Therefore, an ‘asserted Claim’ can be seen as a ‘Claim’ that is being asserted in the development of an assurance case model.

Since the ‘asserted Claim’ means it is the default ‘Claim’, the visual representation of the ‘asserted Claim’ is the visual representation of the ‘Claim’ which is drawn as a rectangle where the claim statement can be written within the rectangle (and an optional element ID can be placed on the top-left of the rectangle when the ‘asserted Claim’ applied the ‘gid’ attribute via inheritance). The visual variables used by the visual representation an ‘asserted Claim’ and its design rationale are also similar to the ‘Claim’ element.

‘needsSupport Claim’

By respecting the inherited design constraint of the ‘needsSupport’, the ‘needsSupport Claim’ visual representation can be created as illustrated in Figure 4.13.

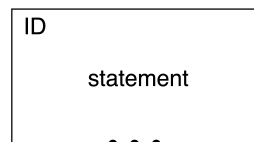


FIGURE 4.13: The example of ‘needsSupport’ when being applied in a Claim (i.e. ‘needsSupport Claim’)

The application of the inherited design constraints of the ‘needsSupport’, in this case is the placement of the three dots on the bottom part of the ‘Claim’, is expected can help to indicate to the notation user that this type of ‘Claim’ is not completed and further argumentation element has yet to be provided to support the ‘Claim’.

Regarding the compositional rule of the ‘needsSupport Claim’, it is generally inherited from the compositional rule of the ‘Claim’: the ‘needsSupport Claim’ can be associated with other ‘Assertion’ type including ‘Claim’. When a ‘needsSupport Claim’ is associated with a ‘Claim’, it must be located either below or on the right/left-side if the supported ‘Claim’. Other elements must not located below the ‘needsSupport Claim’ due to consider the semantic of the ‘needsSupport Claim’ that stated it is

a 'Claim' that is intentionally declared as requiring further evidence or argumentation, therefore, no elements can be located below the 'needsSupport Claim'.

'assumed Claim'

When a 'Claim' is declared as being 'assumed', then the 'Claim' is called as an 'assumed Claim'. By respecting the design constraint of the 'assumed', the 'assumed Claim' visual representation can be created as illustrated in Figure 4.14.

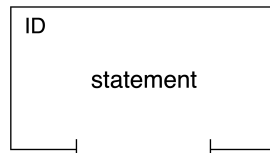


FIGURE 4.14: The visual representation of an 'assumed Claim'

By respecting the design constraints of the 'assumed' enumeration literal of the inherited 'assertion-Declaration' attribute, the visual representation of the 'Claim' is modified: a line gap is placed on the bottom part of the 'Claim' in order to show that this 'Claim' is declared as being assumed without any supporting evidence or argumentation.

The 'assumed Claim' can be associated with the other types of 'Assertion' such as 'Claim'. This compositional rule is inherited from the 'Claim' class. When an 'assumed Claim' is associated with a 'Claim', it must be located either below or on the right/left-side if the supported 'Claim'. Other elements must not be located below the 'assumed Claim' due to consider its semantic that stated it is a 'Claim' that is intentionally declared as being assumed without supporting argument or evidence, therefore, no elements can be located below the 'assumed Claim'.

'axiomatic Claim'

By respecting the design constraint of the 'axiomatic', the 'axiomatic Claim' visual representation can be created. An 'axiomatic Claim' is an instantiation of a 'Claim' when being declared as an axiomatically true by the notation user. Figure 4.15 illustrates the visual representation of the 'axiomatic Claim'.

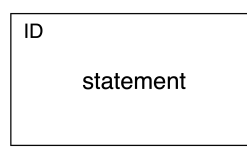


FIGURE 4.15: The visual representation of an 'axiomatic Claim'

By considering the design constraints of the 'axiomatic' that is applied to the 'Claim' visual representation, a thick line is placed below the visual representation of the 'Claim' as can be seen in Figure

4.14. The thick line is used to indicate that there is no need to place any other argumentation element below this ‘Claim’ since it is defined as an ‘axiomatic Claim’.

The compositional rule of the ‘axiomatic Claim’ is inherited from the compositional rule of the ‘Claim’. An ‘axiomatic Claim’ can be associated with an ‘Assertion’ type such as a ‘Claim’. When it is being associated with another ‘Claim’, an ‘axiomatic Claim’ must be located either below or on the right/left-side if the supported ‘Claim’. Other elements must not be located below the ‘axiomatic Claim’ due to its semantic that states it is a ‘Claim’ that is intentionally declared as being axiomatically true so no further argumentation is needed to support this claim, therefore, no elements can be located below the ‘axiomatic Claim’.

‘defeated Claim’

For the ‘Claim’ that is declared as being ‘defeated’, it must be visualised by the representation that can denote the meaning of a ‘Claim’ that is defeated by the counter-argument or evidence. The creation of the ‘defeated Claim’ visual representation can be done by applying the visual inheritance process by combining the visual representation of the ‘Claim’ and the ‘defeated’. The result of the creation of the ‘defeated Claim’ visual representation through the application of the visual inheritance is illustrated in Figure 4.16.

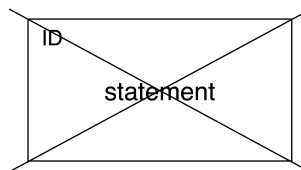


FIGURE 4.16: The visual representation of an ‘defeated Claim’

By applying the design constraints of the ‘defeated’ to the visual representation of the ‘Claim’ (placing a cross on top of the ‘Claim’), the notation user is expected to understand that this ‘Claim’ is defeated by a counter-evidence/argumentation.

Similar with other types of ‘Claim’, the compositional rule of the ‘defeated Claim’ is inherited from the compositional rule of the ‘Claim’: a ‘defeated Claim’ can be associated with an ‘Assertion’ type such as a ‘Claim’. There should be at least an element that must be located below the ‘defeated Claim’ in order to show that the ‘defeated Claim’ is defeated by that element that is connected via ‘Counter inference/evidence’ relationship.

‘asCited Claim’

When the ‘asCited’ enumeration literal is being applied to a ‘Claim’, the ‘Claim’ is called an ‘asCited Claim’ which means that this ‘Claim’ is citing another ‘Claim’. The visual representation of the ‘asCited Claim’ is shown in Figure 4.17.

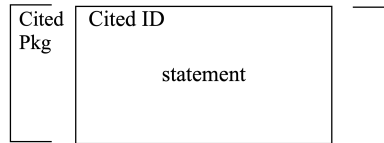


FIGURE 4.17: The visual representation of the ‘asCited Claim’

By applying the design constraints of the ‘asCited’ to the ‘Claim’ visual representation, the notation user is expected to get the meaning that this ‘Claim’ cites another ‘Claim’ where the value (type) of the cited ‘Claim’ is derived to the citing ‘Claim’. For example, if the cited ‘Claim’ is declared as an ‘asserted Claim’, therefore, the value of the citing ‘Claim’ is also an ‘asserted Claim’. The element ID of the cited ‘Claim’ can be located within (top-left) of the rectangle. The location of the cited ‘Claim’ can be placed within (top-left) of the square brackets. The compositional rule of the ‘asCited Claim’ as a ‘Claim’ which cited another ‘Claim’ is following the compositional rule of the cited ‘Claim’.

‘isCitation’

Based on the semantic, ‘isCitation’ can be applied to any sub-classes of the ‘SACMElement’, including the elements of the SACM argumentation metamodel, in order to indicate whether the ‘SACMElement’ cites another ‘SACMElement’. When the ‘isCitation’ is applied to the ‘Claim’, the ‘Claim’ is called as an ‘asCited Claim’. Its visual representation can be seen in Figure 4.17.

‘abstract Claim’

In the case of a ‘Claim’ that is declared as being abstract, this ‘Claim’ is called as an ‘abstract Claim’. An ‘abstract Claim’ is a ‘Claim’ that is declared as part of a pattern or template. The visual representation of an ‘abstract Claim’ is shown in Figure 4.18. The ‘Claim’ is drawn using a dash line in order to respect the design constraint of the ‘isAbstract’.

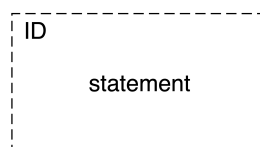


FIGURE 4.18: ‘abstract Claim’ visual representation

The design constraints of the inherited ‘isAbstract’ attribute is applied to the visual representation of the ‘Claim’ that makes the ‘Claim’ visual representation is rendered as a rectangle that is drawn using dash line.

4.6.5 ‘AssertedRelationship’ and its variants

‘AssertedRelationship’ is the sub-class of the ‘Assertion’ class that is defined as an abstract class. According to the SACM specification, the semantics of the ‘AssertedRelationship’ is defined as follows:

"In SACM, the structure of an argument is declared through the linking together of primitive ‘ArgumentAssets’. For example, a sufficient inference can be asserted to exist between two ‘claims’ (“‘Claim’ A implies ‘Claim’ B”) or sufficient evidence can be asserted to exist to support a ‘claim’ (“‘Claim’ A is evidenced by Evidence B”). An inference asserted between two ‘claims’ (A – the source – and B – the target) denotes that the truth of ‘Claim’ A is said to infer the truth of ‘Claim’ B".

Based on the semantics, the ‘AssertedRelationship’ is the abstract association class that enables the ‘ArgumentAssets’ of any structured argument to be linked together. The linking together of ‘ArgumentAssets’ allows the user to declare the relationship that they assert to hold between these elements.

Since the ‘AssertedRelationship’ is an abstract class, its visual representation is not created. However, the semantics of the ‘AssertedRelationship’ is used as a design constraint that can be applied by its concrete sub-classes for the purpose of the creation of the visual representations.

A line is used to represent any instantiations of the ‘AssertedRelationship’ (via its concrete sub-classes). A line is commonly used in visual modelling languages, such as UML, BPMN, SysML, and I*, as concrete syntax to represent a relationship between elements. This design constraint can be inherited and applied by the concrete sub-classes of the ‘AssertedRelationship’: ‘AssertedInference’, ‘AssertedEvidence’, ‘AssertedContext’, ‘AssertedArtifactSupport’, and ‘AssertedArtifactContext’.

The ‘AssertedRelationship’ class has three associations:

- ‘source’: ‘ArgumentAsset’ [1..*] - reference to the ‘ArgumentAsset(s)’ that are the source (starting point) of the relationship.

The ‘source’ association can be used to denote the source element of the application of the ‘AssertedRelationship’. For example, the ‘source’ association is used to indicate the source element that is connected to the ‘AssertedInference’ relationship visual representation.

- ‘target’: ‘ArgumentAsset’ [1..*] - reference to the ‘ArgumentAsset(s)’ that are the target (ending point) of the relationship.

The ‘target’ association can be used to denote the target element of the application of the ‘AssertedRelationship’. For example, the ‘target’ association is used to indicate the target element that is connected to the ‘AssertedInference’ relationship visual representation.

- ‘reasoning’: ‘ArgumentReasoning’ [0..1] – can be used as an optional reference to a description of the reasoning underlying the ‘AssertedRelationship’.

The association of the ‘AssertedRelationship’ can be used as a design constraint that can be inherited and applied by its concrete sub-classes (e.g. ‘AssertedInference’). For example, the ‘source’ and the ‘target’ associations can be used to define that each created visual representation of the concrete sub-classes of the ‘AssertedRelationship’ class must be able to show the source and target element that are connected to the visual representation of the relationship. The application of the ‘target’ association can be a specific head-type of the line that can be used by the visual representation of the concrete sub-classes of the ‘AssertedRelationship’ class. The application of the ‘source’ association can be the line-end that is drawn without a specific type of line-head.

The ‘AssertedRelationship’ class has an attribute named as ‘isCounter’ that can be used to indicate that the AssertedRelationship counters its declared purposes (e.g. counter-evidence when being applied in ‘AssertedEvidence’ relationship). The ‘isCounter’ attribute can be applied by the concrete sub-classes of the ‘AssertedRelationship’ class via inheritance process.

In order to accommodate the ‘isCounter’ attribute, the type of the line-head of the relationship visual representation of the concrete sub-classes of the ‘AssertedRelationship’ must be a 2D-type of line-head, because for the default use of the instantiation of the ‘AssertedRelationship’, the 2D-type of line-head will be drawn using ‘dark’ Brightness, and for the application of the ‘isCounter’, the 2d-type of line-head will be drawn using ‘light’ Brightness. The use of the ‘dark’ Brightness to indicate the default or normal type of relationship is inspired by the default use of supported relationship used in existing notations such as GSN and CAE (although in GSN ‘light’ Brightness used as Context relationship in GSN). In SACM, we used the ‘light’ Brightness as the opposite representation of the ‘dark’ Brightness to indicate ‘Counter’ of the default purpose of the relationship.

Other than the ‘isCounter’ attribute, the ‘AssertedRelationship’ class also has inherited attributes: ‘assertionDeclaration’, ‘gid’, ‘isCitation’, and ‘isAbstract’.

In terms of the application in the visual representation, when considered necessary by the notation user, all of the inherited attributes can also be applied by the concrete sub-classes of the ‘AssertedRelationship’.

To summarise, the following design constraints can be inherited to the sub-classes of the ‘AssertedRelationship’ class:

- The visual representation of the concrete sub-classes of the ‘AssertedRelationship’ must be a line type to denote the semantics of the ‘AssertedRelationship’.
- The line that is used to create the visual representation of the relationship must have a 2D-type of line-head that is placed at a line-end of the relationship that can be used as a pointer to the

targeted element. The other line-end must not have a line-head that can be used to indicate the source element of the relationship (application of the 'target' and 'source' association).

- The 2D-type of line-head must be drawn using 'dark' Brightness to denote the default use of the relationship, and the 'light' Brightness is used to indicate that the relationship is used as a counter-relationship (from its default definition) as the application of the 'isCounter' attribute.
- The created visual representation of the concrete sub-class of the 'AssertedRelationship' should be able to accommodate the other inherited attributes ('assertionDeclaration', 'gid', 'isCitation', and 'isAbstract'). The application of each attribute is following the defined design constraints of the attribute.

The 'AssertedRelationship' has five concrete sub-classes: 'AssertedInference', 'AssertedEvidence', 'AssertedContext', 'AssertedArtifactSupport', and 'AssertedArtifactContext'.

Similar to the 'Claim' element, an 'AssertedRelationship' in SACM such as 'AssertedInference', 'AssertedContext', and 'AssertedEvidence' is also can be declared based on a specific state as defined by the enumeration literal of the 'assertionDeclaration' attribute and the other inherited attributes such as 'isAbstract' that are inherited to the 'AssertedRelationship' class.

As an overview, Figure 4.19 shows the different types of 'AssertedRelationship' in SACM. Similar visual representation decoration as used in visualising different types of 'Claim' is used to visualise different types of 'AssertedRelationship'. For example, an 'assumed AssertedEvidence' is visualised as an arrow headline with a line gap placed in the middle of the line. An 'axiomatic AssertedInference' is visualised as an arrow headline with a thick line placed in the middle of the line. A 'defeated AssertedContext' is visualised as a line with a solid square placed near to a line-end and a cross placed in the middle of the line. The use of similar visual representation to represent the same meaning for different elements, such as a 'Claim' and an 'AssertedRelationship', was designed to provide a visual clue to the notation user.

In Figure 4.19, we also can see a type of relationship that is defined as 'Counter'. This type of relationship can be used to associate a particular element (e.g. 'ArtifactReference' as a reference to evidence) to counter a particular element such as a 'Claim'.

The visual representation of a Counter relationship is visualised as a hollow arrowhead line for an 'AssertedInference' or 'AssertedEvidence', and a hollow square for an 'AssertedContext' relationship. In order to use this relationship in a diagram, we can declare its specific purpose by combining the type of line and the type of the line-head. For example, we can define an 'AssertedInference' relationship by using the Asserted line-type and the Inference head-line-type with a purpose to make an association between 'Claims'. In case we want to define, for example, an 'Asserted Counter-Evidence' relationship, we can use the Asserted line-type and the Counter Evidence line-head type.

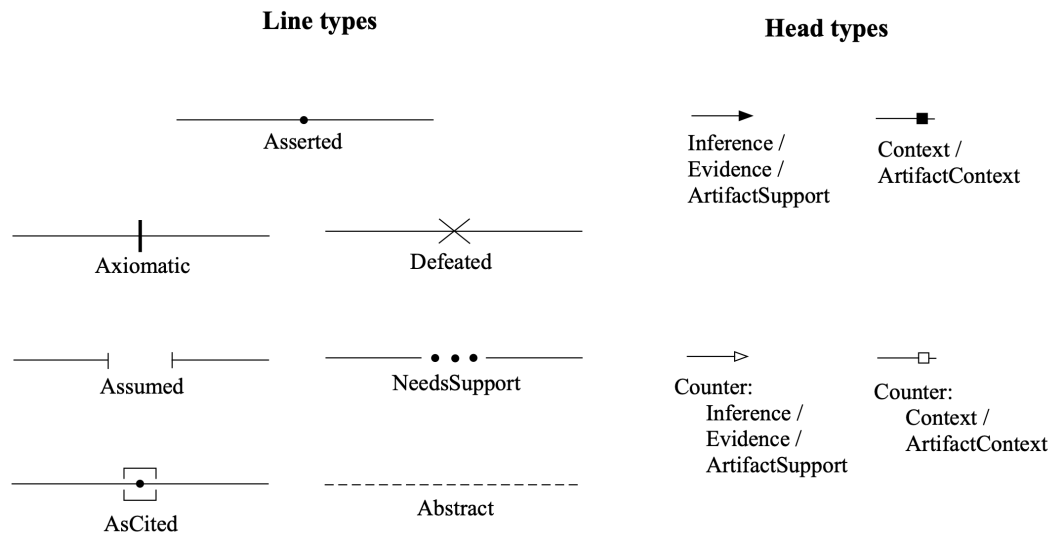


FIGURE 4.19: Types of ‘AssertedRelationship’

The decoration of each type of ‘AssertedRelationship’ is used consistently as used to define a specific type of ‘Claim’ in SACM. This is in order to show family resemblance between these related elements (i.e. to show that these elements has semantic similarities).

4.6.6 ‘AssertedInference’

‘AssertedInference’ is a concrete sub-class of the ‘AssertedRelationship’ class. The semantics of the ‘AssertedInference’ class is defined as follow as stated in the SACM specification:

"The core structure of an argument is declared through the inferences that are asserted to exist between ‘Assertions’ (e.g., ‘Claims’). For example, an ‘AssertedInference’ can be said to exist between two claims (“‘Claim’ A implies ‘Claim’ B”). An ‘AssertedInference’ between two claims (A – the source – and B – the target) denotes that the truth of ‘Claim’ B is said to infer the truth of ‘Claim’ A”.

Based on the semantics, the ‘AssertedInference’ association can be used to records the inference that a user declares to exist between one or more ‘Assertion’ (premise) and another ‘Assertion’ (conclusion). It is important to note that such a declaration is itself an assertion on behalf of the user [9].

Since the ‘AssertedInference’ is a concrete class, therefore, its visual representation needs to be created. The creation of the ‘AssertedInference’ visual representation considered the design constraints that is defined by its super-class (i.e. ‘AssertedRelationship’). In this case, the created visual representation needs to consider the following inherited design constraints:

- The visual representation must be a line type to denotes the semantics of an ‘AssertedRelationship’ type.

- The line that is used to create the visual representation of the relationship must have a 2D-type of line-head that is placed at a line-end of the relationship that can be used as a pointer to the targeted element. The other line-end must not have a line-head that can be used to indicate the source element of the relationship (application of the ‘target’ and ‘source’ association).
- The 2D-type of line-head must be drawn using ‘dark’ Brightness to denote the default use of the relationship, and the ‘light’ Brightness is used to indicate that the relationship is used as a counter-relationship (from its default definition) as the application of the ‘isCounter’ attribute.
- The created visual representation should be able to accommodate the other inherited attributes (‘assertionDeclaration’, ‘gid’, ‘isCitation’, and ‘isAbstract’).

By considering the inherited design constraints, the created visual representation of the ‘AssertedInference’ relationship is shown in Figure 4.20.

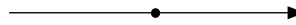


FIGURE 4.20: ‘AssertedInference’ visual representation

‘AssertedInference’ is rendered as a ‘dark’ brightness arrowhead line with a dot placed in the middle of the line as a connection point and represents the actual ‘AssertedRelationship’ object. The edge without an arrow represents the connection point to the source element of the ‘AssertedInference’, and the edge with an arrow represents the connection point to the target element of the ‘AssertedInference’. When used in a diagram, ‘AssertedInference’ must be drawn vertically relative to its source and target elements. The direction of the ‘AssertedInference’ relationship starts from the supporting element to the supported element. A connected dot is used as a connection point when more than one ‘AssertedInference’ are connected. Permitted connections of the ‘AssertedInference’ is defined as follow:

- From (source element) : ‘Claim’ type
- To (target element) : ‘Assertion’ type (e.g. ‘Claim’)

Other than considering the inherited design constraints from the super-class of the ‘AssertedInference’ class, the creation of the ‘AssertedInference’ relationship also considered the existing notation design aspect.

The ‘AssertedInference’ visual representation is inspired from the GSN SupportedBy relationship because the application of them when being used in a diagram is relatively similar, specifically, in the context of as a relationship that can be used to associate between ‘Claims’. In GSN, the SupportedBy relationship mainly used as an inferential or evidential relationship in the argument. When being used in a diagram, an inferential SupportedBy relationship can be identified by the elements that

are connected to the SupportedBy relationship. Those elements must be GSN Goals (or equals to a 'Claim' in SACM).

In SACM, the 'AssertedInference' visual representation is used to connect a 'Claim' (as the supporting element) with an 'Assertion' such as 'Claim' (as the target element). The main difference between the application of the 'AssertedInference' and the GSN SupportedBy relationship is the direction of the arrowhead-line. In GSN, the arrow-head line start from the supported element to the supporting element (the arrowhead-line pointing to the supporting element). Meanwhile, in SACM, the arrowhead-line of 'AssertedInference' relationship start from the supporting element to the supported (targeted) element (the arrowhead-line pointing to the supported element).

For the notation user who have prior knowledge related to the GSN visual representation needs to be careful when using the 'AssertedInference' visual representation due to the difference use of the arrowhead-line direction. Another difference between the GSN SupportedBy relationship and the SACM 'AssertedInference' is related to the used of a dot that is placed in the middle of the SACM 'AssertedInference' visual representation. The dot is only used in SACM 'AssertedInference', meanwhile the GSN SupportedBy is not using this dot as a connection point.

Beside considering existing notation in the creation of the 'AssertedInference' visual representation, an input from the notation stakeholder is also considered. The previous design of the 'AssertedInference' relationship is shown in Figure 4.21. It was rendered as an arrowhead line *without* a dark dot placed in the middle of the line.



FIGURE 4.21: Previous design of the 'AssertedInference' visual representation - without dot in the middle of the line

In the design phase, we were suggested to add a dot in the middle of the line of the 'AssertedInference' relationship (and other asserted-type relationship) by the tool developer, as one of the notation stakeholder, and supported by the other notation stakeholders (e.g., member of committee that has interest in the notation), so it can be applied in a software tool - moreover the dot is considered necessary to be used as a connection point when the 'AssertedInference' is connected with other element (e.g. 'AssertedContext' relationship) when applied in a software tool.

By considering notation stakeholders input, a dot was added to the visual representation of the 'AssertedInference' relationship and other types of 'asserted' relationship such as 'AssertedEvidence' and 'AssertedContext'. With a dot as part of the design of the 'asserted' type relationships, the visual representation of the 'AssertedInference' and other types 'asserted' type relationships should be able to be implemented in a tool support (further evaluation is needed to reduce subjectivity).

The visual variables used and the other design rationale aspects considered in the creation of the 'AssertedInference' visual representation are presented as a visual representation design identity table

that can be read further in Appendix A.2.1 ‘AssertedInference’.

There are several attributes that is inherited to the ‘AssertedInference’ class: ‘assertionDeclaration’, ‘gid’, ‘isCitation’, ‘isAbstract’, and ‘isCounter’.

The application of the ‘assertionDeclaration’ attribute in the ‘AssertedInference’ class is based on the enumeration literals of the ‘assertionDeclaration’ attribute.

Figure 4.32 provides an overview of different types of ‘AssertedInference’ relationship. Each type of ‘AssertedInference’ used the same decoration with each ‘Claim’ types. This is due to show family resemblance among related visual representations. For example, ‘defeated AssertedInference’ used the same decoration as ‘defeated Claim’ (i.e. cross on top of the visual representation) to consistently denote that those elements are declared defeated.

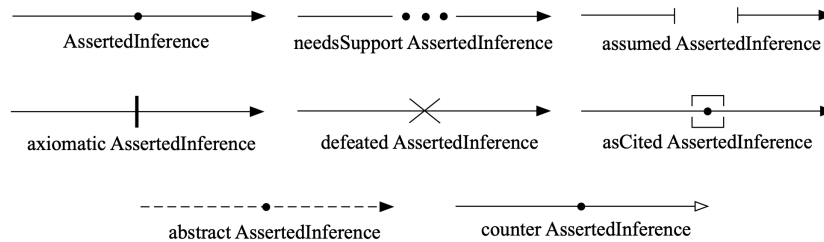


FIGURE 4.22: Types of ‘AssertedInference’ relationship

Each type of ‘AssertedInference’ is created based on visual inheritance by respecting the inherited design constraints of the ‘AssertedInference’ and a specific attribute value such as ‘assumed’ as an enumeration literal of ‘AssertionDeclaration’ attribute. The compositional rules of each type of ‘AssertedInference’ are also inherited from the ‘AssertedInference’ compositional rules when used in a diagram. Visual representation identity of each type ‘AssertedInference’ can be read further in Appendix A.2. ‘AssertedInference’ and its variants.

Theoretically, the ‘gid’ attribute can be applied in the ‘AssertedInference’ class because of the generalisation relationship that is connected from the ‘SACMElement’ to the ‘AssertedInference’ class in the SACM metamodel. If the notation user considered it is necessary for the ‘AssertedInference’ relationship to have a unique element ID, it is can be placed near the middle of the visual representation of the ‘AssertedInference’. This definition is inherited from the ‘gid’ attribute Location/Position visual variable as defined in the previous ‘gid’ sub-section. However, based on the literature, the element ID is commonly used only for the 2D-type visual representation. An element ID rarely found implemented in 1D-type visual representation such as used by a line-type visual representation.

The ‘isCitation’ attribute can be applied in the ‘AssertedInference’ class. Based on the semantics, the application of the ‘isCitation’ attribute to the ‘AssertedInference’ is similar with the application of the ‘asCited AssertedInference’.

The 'isCounter' attribute can be applied in the 'AssertedInference' class. It can be used to indicate that the 'AssertedInference' counters its declared purposes. So, when the 'isCounter' is being applied, the 'AssertedInference' becomes the 'Counter-AssertedInference'. Hence, the arrowhead of the 'AssertedInference' must be drawn as light brightness to respect the inherited design constraints of 'isCounter' attribute.

The 'isAbstract' attribute can be applied in the 'AssertedInference' class that can be used to indicate that the 'AssertedInference' is declared as an abstract element that can be used, for example, as part of a pattern argumentation model. When the 'isAbstract' attribute is applied in the 'AssertedInference' class, it is called as an 'abstract AssertedInference'. The visual representation of an 'abstract AssertedInference', as can be seen in Figure 4.32, is similar with the visual representation of the 'AssertedInference'. The only difference is the line of the 'AssertedInference' is drawn using dash-line texture to respect the inherited design constraints of the 'isAbstract' attribute.

In order to support assurance case pattern development, an 'abstract AssertedInference' can also be declared into a specific type, e.g., for the purpose of the instantiation of element in a model, that can be described as follow:

- An 'abstract AssertedInference' can be declared as an **optional** 'abstract AssertedInference' which mean it is an optional relationship between SACM elements (i.e. could be zero or one). A keyword «optional» is attached around the middle of the 'abstract AssertedInference' relationship to indicate that it is an **optional** 'abstract AssertedInference' relationship as illustrated in Figure 4.23.

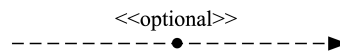


FIGURE 4.23: Optional type of 'abstract AssertedInference' visual representation

- An 'abstract AssertedInference' can be declared as a **multiple** 'abstract AssertedInference' which mean it is a multiple type relationship between SACM elements (i.e. could be zero or more). A keyword «multiple» is attached around the middle of the 'abstract AssertedInference' relationship to indicate that it is a **multiple** 'abstract AssertedInference' relationship as illustrated in Figure 4.24.

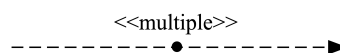


FIGURE 4.24: Multiple type of 'abstract AssertedInference' visual representation

- An 'abstract AssertedInference' can be declared as a **choice** 'abstract AssertedInference' which mean it represents 1-of-n and m-of-n selection of the relationship, a textual annotation indicating the nature of the choice need to be added near the relationship. A keyword «choice» is attached around the middle of the 'abstract AssertedInference' relationship to indicate that it is a **choice** 'abstract AssertedInference' relationship as illustrated in Figure 4.25.

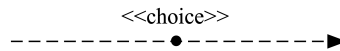


FIGURE 4.25: Choice type of 'abstract AssertedInference' visual representation

Each type of 'abstract'-relationship as implemented in 'abstract AssertedInference' is also applicable for the other types of 'AssertedRelationship' such as 'AssertedEvidence' and 'AssertedContext'. The application example of the 'abstract AssertedInference' is described in the last section in this chapter.

4.6.7 'ArtifactReference'

'ArtifactReference' is a concrete sub-class of the 'ArgumentAsset' class. According to the SACM specification document, the semantics of the 'ArtifactReference' class is described as follow:

"It is necessary to be able to cite artifacts that provide supporting evidence, context, or additional description within an argument structure. 'ArtifactReferences' allow there to be an objectified citation of this information within the structured argument, thereby allowing the relationship between this artifact and the argument to also be explicitly declared".

Based on the semantics, 'ArtifactReference' enables the citation of an artifact as information that relates to or support the structured argument. Since the 'ArtifactReference' is a concrete class, its visual representation needs to be created.

The visual representation of the 'ArtifactReference' is created by considering the semantic transparency principle which is the created visual representation should suggest or represent its semantic meaning. An 'ArtifactReference' is denoted using a document icon with an arrow pointing out from the icon (to denote a reference) which is placed on the top right of the icon. The 'ArtifactReference' statement can be written within the icon and can be stated as noun-phrases. The unique element identifier can be placed at the top-left corner of the icon in the case the 'gid' attribute is applied in the 'ArtifactReference'.

Figure 4.26 illustrates the visual representation of the 'ArtifactReference'.

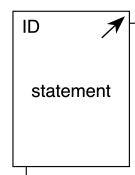


FIGURE 4.26: 'ArtifactReference' visual representation

The visual representation of 'ArtifactReference' is not inherited the visual representation of other elements. In the creation of the 'ArtifactReference' visual representation, the design of the existing notation is considered.

The visual representation of the ‘ArtifactReference’ that is shown in Figure 4.26 is the revised version of the ‘ArtifactReference’ visual representation. The previous version of the ‘ArtifactReference’ visual representation is illustrated in Figure 4.27.

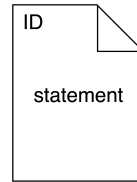


FIGURE 4.27: Previous version of the ‘ArtifactReference’ visual representation

Due to consider the similarity with existing notations (e.g. UML Comment and SysML Comment visual representation) and to avoid a visual representation clash with existing notations, the previous version of the ‘ArtifactReference’ visual representation (as shown in Figure 4.27) is revised based on the input from the OMG committee in the development phase of the SACM argumentation notation.

As for the Semantic transparency, we considered the ‘ArtifactReference’ visual representation as an icon-type visual representation because its appearance may suggest or represents its semantic meaning. Regarding the visual variables used and other design rationales behind the creation of the ‘ArtifactReference’ are available in Appendix A.3.

Related the compositional rule of the ‘ArtifactReference’, in SACM, the citation of an artifact as a contextual or evidential information that relates to the structured argument can be done using the ‘ArtifactReference’. When applied in a diagram, the ‘ArtifactReference’ that is used either as a reference to a contextual information or evidential information can be identified by its position and the relationship type that connect the ‘ArtifactReference’ and its targeted element.

When an ‘ArtifactReference’ is used as a reference to an evidential information, it must be located below the target element:

- If the target element is a ‘Claim’, the relationship between them must be declared using an ‘AssertedEvidence’.
- If the target element is another ‘ArtifactReference’ (as a reference to context), the relationship between them must be declared using an ‘AssertedArtifactSupport’.

When an ‘ArtifactReference’ is used as a reference to a contextual information, it must be located horizontally (right/left side) relative to its target element:

- If the target element is a ‘Claim’, the relationship between them must be declared using an ‘AssertedContext’.

- If the target element is another ‘ArtifactReference’ (as a reference to evidence), the relationship between them must be declared using an ‘AssertedArtifactContext’.

The ‘ArtifactReference’ has no sub-classes. It has attributes that are inherited from the ‘SACMElement’ class: ‘gid’, ‘isCitation’, and ‘isAbstract’. The application of the ‘gid’ attribute has been previously described in the creation of the ‘ArtifactReference’ visual representation.

As for the ‘isCitation’ and ‘isAbstract’ attributes, they can be applied in the ‘ArtifactReference’ class by respecting the design constraints of each attribute.

‘abstract ArtifactReference’

‘abstract ArtifactReference’ is the application of the inherited ‘isAbstract’ attribute in the ‘ArtifactReference’ class. An ‘abstract ArtifactReference’ can be described as an ‘ArtifactReference’ that is declared as part of an argument pattern or template. It can be instantiated when being applied in more concrete and specific cases. The application of the ‘isAbstract’ attribute in the ‘ArtifactReference’ class is for the purpose to support the argument pattern development in assurance cases.

The creation of the ‘abstract ArtifactReference’ visual representation is based on visual inheritance of the ‘ArtifactReference’ visual representation and respecting the design constraints of the ‘isAbstract’ attribute. The created ‘abstract ArtifactReference’ visual representation is illustrated in Figure 4.28.

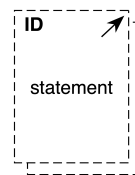


FIGURE 4.28: ‘abstract ArtifactReference’ visual representation

An ‘abstract ArtifactReference’ visual representation is created to be similar with the ‘ArtifactReference’ visual representation since it is inherited the visual representation of the ‘ArtifactReference’. The only difference is related to the line texture type that form the visual representation of the ‘abstract ArtifactReference’ (except for the arrowhead line). The line texture in this case is using the dash line type. The used of the dash line can be seen as an effort to comply and respecting the inherited design constraint of the ‘isAbstract’ attribute.

The compositional rules of the ‘abstract ArtifactReference’ visual representation when used in a diagram are inherited from the ‘ArtifactReference’. Related to the visual variables used and the other design rationale of ‘abstract ArtifactReference’, they can be found in Appendix A.3.

‘asCited ArtifactReference’

In the case if the 'isCitation' attribute is applied to the 'ArtifactReference', that means the 'ArtifactReference' cites another 'ArtifactReference' element. The 'ArtifactReference' is not inheriting the 'assertionDeclaration' attribute from 'Assertion' class. However, we called an 'ArtifactReference' that cites another 'ArtifactReference' element as an 'asCited ArtifactReference' in order to have a consistent naming with the application of the 'Claim' that cites another 'Claim' element that is called as an 'asCited Claim'.

In terms of visual representation, an 'asCited ArtifactReference' is visually represented using the 'ArtifactReference' visual representation that is placed between the square brackets (as shown in Figure 4.29) to indicates that the 'ArtifactReference' is citing another 'ArtifactReference' element.

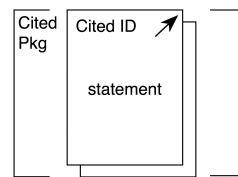


FIGURE 4.29: 'asCited ArtifactReference' visual representation

The visual representation of the 'asCited ArtifactReference' is created by respecting the design constraints of the 'ArtifactReference' and 'isCitation' attribute. The visual variables used and the design rationale aspects related to the creation of the 'asCited ArtifactReference' can be found in the Appendix A.3.

Similar with the 'abstract ArtifactReference' element, the compositional rules of the 'asCited ArtifactReference' are inherited the compositional rules from the 'ArtifactReference' element.

4.6.8 'AssertedEvidence'

'AssertedEvidence' is the concrete sub-class of the 'AssertedRelationship' class. Its semantics is described as follow based on the SACM specification document:

"Where evidence (cited by 'ArtifactReference') exists that helps to establish the truth of a 'Claim' in the argument, this relationship between the 'Claim' and the evidence can be asserted by an 'AssertedEvidence' association. An 'AssertedEvidence' association between an artifact cited by an 'ArtifactReference' and a 'Claim' (A – the source evidence cited – and B – the target claim) denotes that the evidence cited by A is said to help establish the truth of 'Claim' B".

Based on the semantics, 'AssertedEvidence' association can be used to records the declaration that one or more artifacts of Evidence (cited by 'ArtifactReference') provide information that helps establish the truth of a 'Claim'. It is important to note that such a declaration is itself an assertion on behalf of the user. The artifact (cited by an 'ArtifactReference') may provide evidence for more than one 'Claim'.

Since the ‘AssertedEvidence’ class is a concrete class, its visual representation needs to be created. The creation of the ‘AssertedEvidence’ visual representation is respected the inherited design constraints of its super-class (i.e. ‘AssertedRelationship’). By considering the inherited design constraints, the created visual representation of the ‘AssertedEvidence’ is illustrated in Figure 4.30.



FIGURE 4.30: ‘AssertedEvidence’ visual representation

An ‘AssertedEvidence’ is rendered as a dark brightness arrowhead-line with a dot placed in the middle of the line that can be used as a connection point and represents the actual ‘AssertedRelationship’ object. The edge without an arrow represents the connection point to the source element of the ‘AssertedEvidence’, and the edge with an arrow represents the connection point to the target element of the ‘AssertedEvidence’. When used in the diagram, ‘AssertedEvidence’ must be drawn vertically relative to its source and target elements.

As shown in Figure 4.30, the visual representation of the ‘AssertedEvidence’ looks identical with the visual representation of the ‘AssertedInference’ relationship. The previous design of the ‘AssertedEvidence’ visual representation is illustrated in Figure 4.31.

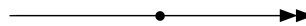


FIGURE 4.31: Previous version of the ‘AssertedEvidence’ visual representation

In the notation development phase, we considered the input from the notation stakeholder such as OMG committee who have interest in the development of the SACM notation. Due to considering the graphic complexity of a notation (i.e. managing the number of the visual representations in a notation [3]) and avoid confusion related to the used of the double arrowhead, the visual representation of the ‘AssertedEvidence’ is decided to be designed similar with the ‘AssertedInference’ relationship. They can be distinguished when being used or presented in a diagram. The direction of the ‘AssertedEvidence’ relationship starts from the evidence (cited by ‘ArtifactReference’) to the supported element, or in other words, the source of ‘AssertedEvidence’ relationship must be ‘ArtifactReference’ (as evidence) as defined as the constraint described in SACM specification [9]. The position of the ‘ArtifactReference’ as evidence must be located below the supported element.

Permitted connections of the ‘AssertedEvidence’:

- From (source element): ‘ArtifactReference’ (as evidence)
- To (target element): ‘Claim’

By identifying the types of element as the source and target of the relationship, the notation user can identify the differentiation of the application of the ‘AssertedEvidence’ and the ‘AssertedInference’ relationship.

Similar with the ‘AssertedInference’ class, the ‘AssertedEvidence’ is also inherited several attributes: ‘assertionDeclaration’, ‘gid’, ‘isCitation’, ‘isAbstract’, and ‘isCounter’.

In general, the visual representation creation of the ‘AssertedEvidence’ relationship that applies a specific type of inherited attribute is similar with the creation of the visual representation of the ‘AssertedInference’ relationship that applies a specific type of inherited attribute. The resulting visual representations are also looks similar. They can be differentiated by identifying the source and target elements that are connected to them. The compositional rules of each type ‘AssertedEvidence’ relationship (e.g. ‘assumed AssertedEvidence’, ‘axiomatic AssertedEvidence’, and ‘defeated AssertedEvidence’) are inherited the compositional rules of the ‘AssertedEvidence’ relationship.

As a summary, Figure 4.32 illustrates different types of ‘AssertedEvidence’ relationship.

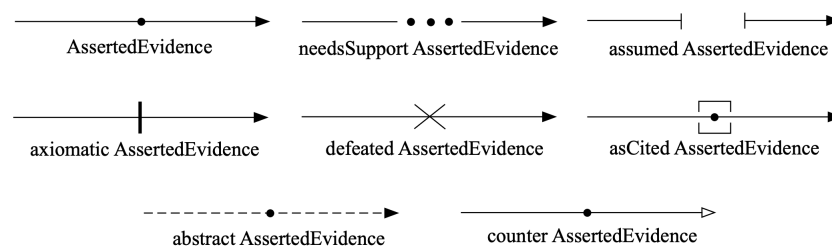


FIGURE 4.32: Types of ‘AssertedEvidence’ relationship

As can be seen in Figure 4.32, each type of ‘AssertedEvidence’ used the same decoration with each type of the ‘AssertedInference’ and ‘Claim’. This is, again, due to show family resemblance among related visual representations. For example, ‘defeated AssertedEvidence’, ‘defeated AssertedInference’, ‘defeated Claim’. They are using the same decoration (a cross placed on top of the visual representation) to consistently denotes that the element is defeated by counter-evidence/argumentation. Regarding the design rationale of each type ‘AssertedEvidence’ relationship can be found in Appendix A.4.

4.6.9 ‘AssertedContext’

‘AssertedContext’ is a concrete sub-class of the ‘AssertedRelationship’ class. Its semantics is described as follow according to the SACM specification document:

“Contextual information often needs to be cited in order to make clear the interpretation and scope of a ‘Claim’. For example, a ‘Claim’ can be said to be valid only in a defined context (“‘Claim’ A is asserted to be true only in a context as defined by the information cited by Artifact B” or conversely “InformationItem B is the asserted context for ‘Claim’ A”). Contextual ‘Claims’ often need to be cited as preconditions for an ‘Assertion’. For example, a ‘Claim’ may be asserted only in the context of another ‘claim’ (“‘Claim’ A is asserted to be true only in a context where ‘Claim’ B is true”).”

Based on the semantics, 'AssertedContext' can be used to declare that the artifact cited by an 'ArtifactReference' provides the context for the interpretation and scoping of a 'Claim' element. In addition, the 'AssertedContext' can be used to declare a 'Claim' asserted as necessary context (i.e. a precondition) for another 'Assertion' such as a 'Claim'.

Since the 'AssertedContext' is a concrete class, its visual representation needs to be created. The creation of the 'AssertedContext' visual representation is respected the inherited design constraints of its super-class (i.e. 'AssertedRelationship'). By considering the inherited design constraints, the visual representation of the 'AssertedContext' is created as shown in Figure 4.33.

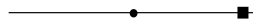


FIGURE 4.33: 'AssertedContext' visual representation

An 'AssertedContext' is rendered as a line with a solid square located near of a line-end (used as the target end) and a dot placed in the middle of the line as a connection point and represents the actual asserted object. The edge without a solid square represents the connection point to the source element of the 'AssertedContext', and the edge with a solid square represents the connection point to the target element of the 'AssertedContext'.

When used in a diagram, the direction of the 'AssertedContext' relationship starts from the supporting element, in this case either an 'ArtifactReference' (as a Context) or a 'Claim' element that is used as preconditions for a particular 'Assertion' such as a 'Claim, to the supported element where in this case is a 'Claim' element. An 'AssertedContext' must be drawn horizontally relative to the source and target element when being used in a diagram. A connected dot that is placed in the middle of the 'AssertedContext' relationship is used as a connection point when more than one 'AssertedContext' are connected. Permitted connections of the 'AssertedContext' is summarised as follow:

- From (source element) : 'ArtifactReference' (as a Context) or a 'Claim' element (as a necessary context (i.e. a precondition))
- To (target element) : 'Assertion' type (e.g. 'Claim').

In the creation of the 'AssertedContext' visual representation, the input from the notation stakeholder, in this case is the OMG committee who have interest with the SACM notation, is also considered.

The previous design of the 'AssertedContext' is rendered as a line with a diamond head (drawn as 'dark' Brightness - used as a pointer to the targeted element). As shown in Figure 4.34, we considered this previous design is too similar with the UML Composition relationship that might confuse the notation user who familiar with UML notation. In other words, a visual representation clash with existing notation might occurred if the 'AssertedContext' is visually represented using the diamond-head

line due to the semantics differences between the SACM ‘AssertedContext’ and the UML Composition.

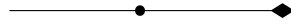


FIGURE 4.34: Previous design of the ‘AssertedContext’ visual representation

Considering the input from the notation stakeholder in the design phase, the visual representation of the ‘AssertedContext’ is being revised. The revised design of the ‘AssertedContext’ that also has been approved by the OMG committee is shown in Figure 4.33.

The current visual representation of the ‘AssertedContext’ is not adopted from any existing notations. The notation clash with existing notations is considered avoided.

We also considered the tool support implementation factor in the creation of the ‘AssertedContext’ relationship. Similar to the ‘AssertedInference’ and ‘AssertedEvidence’ relationship, the visual representation of the ‘AssertedContext’ relationship previously designed without a dot was placed in the middle of its line. The notation stakeholders such as tool developer suggested that a dot need to be added in the middle of the ‘AssertedContext’ relationship so it can be applied in a software tool as a connection point when the ‘AssertedContext’ relationship is connected with other element, e.g. ‘AssertedEvidence’ relationship. Due to consider the input from the notation stakeholders, a dot was added as part of the ‘AssertedContext’ relationship visual representation and it should be able to be applied in a software tool.

The ‘AssertedContext’ has no sub-classes and no association relationship. The ‘AssertedContext’ has inherited the following attributes: ‘assertionDeclaration’, ‘gid’, ‘isCitation’, ‘isAbstract’, and ‘isCounter’.

In general, the visual representation creation of the ‘AssertedContext’ relationship that applies a specific type of inherited attribute is similar with the creation of the visual representation of the ‘AssertedInference’ and ‘AssertedEvidence’ relationship that applies a specific type of inherited attribute. Figure 4.32 provides an overview of different types of ‘AssertedContext’ relationship. Each type of ‘AssertedContext’ used the same decoration as used by the different types of ‘AssertedInference’ and ‘AssertedEvidence’. This is due to show family resemblance among related visual representations. For example, ‘assumed AssertedContext’ used the same decoration as ‘assumed AssertedInference’ and ‘assumed AssertedEvidence’ (i.e. line gap placed in the middle of the line) to consistently denote that those elements are declared as an assumption.

Each type of ‘AssertedContext’ is created based on visual inheritance by respecting the inherited design constraints of the ‘AssertedContext’ and a specific attribute value such as ‘assumed’ as an enumeration literal of ‘AssertionDeclaration’ attribute. The compositional rules of each type of ‘AssertedContext’ are also inherited from the ‘AssertedContext’ compositional rules when used in a diagram. Visual representation identity of each type ‘AssertedContext’ can be read in Appendix A.5.

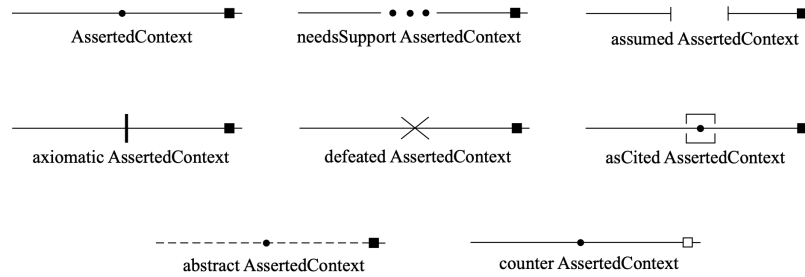


FIGURE 4.35: Types of ‘AssertedContext’ relationship

4.6.10 ‘AssertedArtifactSupport’

The ‘AssertedArtifactSupport’ is a concrete sub-class of the ‘AssertedRelationship’ class. According to SACM specification, the semantics of the ‘AssertedArtifactSupport’ is described as follow:

"The truth of the assertions associated with an artifact are supported by the assertions that are associated with one or more other artifacts. Note: this can be an ambiguous relationship if the nature of these Assertions is unclear. In such cases, it would be clearer to declare explicit ‘AssertedInferences’ between ‘Claims’ drawn out from the ‘ArtifactReference’".

An ‘AssertedArtifactSupport’ is a relationship that can be used to record the assertion that one or more artifacts support another artifact, in this context, the source and target of ‘AssertedArtifactSupport’ must be of type ‘ArtifactReference’ (as described in SACM specification - the constraints of the ‘AssertedArtifactSupport’)[9].

Since the ‘AssertedArtifactSupport’ is a concrete class, its visual representation needs to be created. We considered the semantics and the constraints of the ‘AssertedArtifactSupport’ to create the visual representation of the ‘AssertedArtifactSupport’ and its compositional rules when used in a diagram. The inherited design constraints from the super-class of the ‘AssertedArtifactSupport’ class (i.e. ‘AssertedRelationship’) are also considered in creating the ‘AssertedArtifactSupport’ relationship.

By considering the semantics and the inherited design constraints, the created visual representation of the ‘AssertedArtifactSupport’ is illustrated in Figure 4.36.

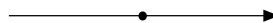


FIGURE 4.36: ‘AssertedArtifactSupport’ visual representation

The visual representation of the ‘AssertedArtifactSupport’ is rendered as an arrowhead line (drawn as a ‘dark’ brightness) with a dot placed on the middle of the line that can be used as a connection point and presents the actual asserted object. The line-end with an arrowhead is used as a pointer to be connected to the targeted element. The line-end without an arrowhead is used as an indicator for the source element of the ‘AssertedArtifactSupport’. When used in a diagram, ‘AssertedArtifactSupport’ must be drawn vertically relative to its source and target elements.

As shown in Figure 4.36, the visual representation of the ‘AssertedArtifactSupport’ looks identical with the visual representation of the ‘AssertedInference’ and ‘AssertedEvidence’. Besides considering the design constraints in creating the ‘AssertedArtifactSupport’, the graphic complexity design principle is also considered to create the ‘AssertedArtifactSupport’ visual representation. In order to manage the number of the visual representations in a notation, the ‘AssertedArtifactSupport’ is designed to be identical with the ‘AssertedInference’ and ‘AssertedEvidence’ relationship with the following permitted connections specifically applies only for the ‘AssertedArtifactSupport’ relationship:

- The source element must be: ‘ArtifactReference’ (as evidence) and located vertically (below) relative to the targeted element.
- The target element must be: ‘ArtifactReference’ (can be as a reference to a contextual information or a reference to another evidential information).

We also considered the tool support implementation in the creation of the ‘AssertedArtifactSupport’ visual representation. A dot was placed in the middle of the line as part of the design of the ‘AssertedArtifactSupport’ relationship. The ‘AssertedArtifactSupport’ relationship was previously designed without a dot as part of its visual representation. Due to consider the notation stakeholders input such as tool developer, a dot was added so the ‘AssertedArtifactSupport’ relationship can be applied in a software tool specifically to be used as a connection point to connect the ‘AssertedArtifactSupport’ relationship with other elements such as ‘AssertedContext’. By considering the tool developer input in the design of the ‘AssertedArtifactSupport’ visual representation, the ‘AssertedArtifactSupport’ relationship element should be able to be applied in a software tool.

The ‘AssertedArtifactSupport’ class has no sub-classes. It is inherited several attributes that can be applied to its visual representation: ‘assertionDeclaration’, ‘gid’, ‘isCitation’, ‘isAbstract’, and ‘isCounter’.

Generally, the visual representation creation of the ‘AssertedArtifactSupport’ relationship that applies a specific type of inherited attribute is similar with the creation of the visual representation of the ‘AssertedInference’ or ‘AssertedEvidence’ relationship that applies a specific type of inherited attribute. The resulting visual representations are also looks similar. They can be differentiated by identifying the source and target elements that are connected to them. The compositional rules of each type ‘AssertedArtifactSupport’ relationship (e.g. ‘needsSupport AssertedArtifactSupport’, ‘axiomatic AssertedArtifactSupport’, and ‘defeated AssertedArtifactSupport’) are inherited the compositional rules of the ‘AssertedArtifactSupport’ relationship.

As a summary, Figure 4.37 illustrates different types of ‘AssertedArtifactSupport’ relationship.

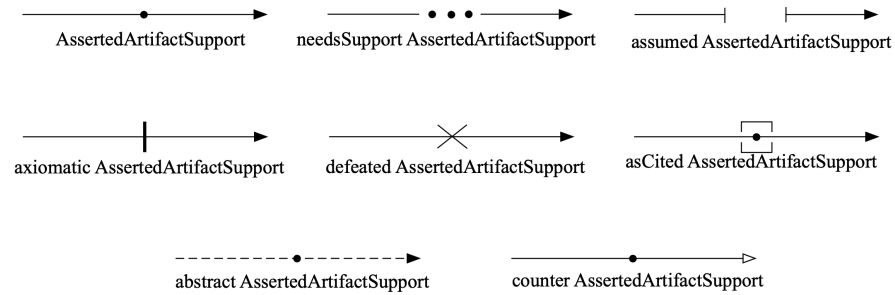


FIGURE 4.37: Types of ‘AssertedArtifactSupport’ relationship

As can be seen in Figure 4.37 similar design decoration as used in creating each type of, for example ‘AssertedInference’, is used to represent each type of ‘AssertedArtifactSupport’. This is done to show family resemblance among related elements. Regarding the design rationale of each type of ‘AssertedArtifactSupport’ relationship can be found in Appendix A.6.

4.6.11 ‘AssertedArtifactContext’

‘AssertedArtifactContext’ is a concrete sub-class of the ‘AssertedRelationship’ class. Its semantics is described as follows according to the SACM specification document:

“One or more other artifacts provide the necessary context in which the assertions associated with another artifact should be understood. Note: this can be an ambiguous relationship if the nature of these Assertions is unclear. In such cases, it would be clearer to declare explicit ‘AssertedContext’ between Claims drawn out from the ‘ArtifactReference’”.

‘AssertedArtifactContext’ can be used to record the assertion that one or more artifacts provide context for another artifact. The use of the ‘AssertedArtifactContext’ must follow the following constraint as defined in SACM specification:

“The source and target of ‘AssertedArtifactContext’ must be of type ‘ArtifactReference’”.

Since the ‘AssertedArtifactContext’ is a concrete class, its visual representation needs to be created by considering its semantics, constraint of use, and inherited design constraints from the super-class of the ‘AssertedArtifactContext’. The created visual representation of the ‘AssertedArtifactContext’ is illustrated in Figure 4.38.

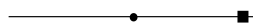


FIGURE 4.38: ‘AssertedArtifactContext’ visual representation

An ‘AssertedArtifactContext’ denotes as a line with a dark brightness square located near of a line-end that is used as a pointer to indicate the targeted element of the ‘AssertedArtifactContext’ relationship. The other line-end is used as to indicate the source element of the ‘AssertedArtifactContext’ relationship. A dot is placed in the middle of the line that can be used as a connection point and represents the

actual asserted object. When used in a diagram, ‘AssertedArtifactContext’ must be drawn horizontally relative to its source and target elements.

The visual representation of the ‘AssertedArtifactContext’ relationship looks identical with the visual representation of the ‘AssertedContext’ relationship. Other than considering the semantics, constraints of use, and the inherited design constraints, the creation of the ‘AssertedArtifactContext’ visual representation also considered the graphic complexity design principle: due to manage the number of visual representations in a notation, the ‘AssertedArtifactContext’ visual representation is created identical with the ‘AssertedContext’ relationship.

We utilised the ‘AssertedArtifactContext’ constraint of use to distinguish the ‘AssertedArtifactContext’ and ‘AssertedContext’ relationship. So, the compositional rules of the ‘AssertedArtifactContext’ can be summarised as follow:

- The source element must be of type ‘ArtifactReference’ as a reference to a contextual information.
- The target element must be of type ‘ArtifactReference’ as a reference to a evidential information.
- The ‘AssertedArtifactContext’ relationship must be drawn horizontally relative to its source and target elements due to consider design consistency between ‘AssertedContext’ and ‘AssertedArtifactContext’ (both of them can be used for the purpose of providing context to SACM element).

The ‘AssertedArtifactContext’ has no sub-classes. It has inherited attributes that can be summarised as follow: ‘assertionDeclaration’, ‘gid’, ‘isCitation’, ‘isAbstract’, and ‘isCounter’.

The creation of the visual representations of each type of the ‘AssertedArtifactContext’ that applied the inherited attributed is relatively similar with the creation of the ‘AssertedContext’ visual representation types. The produced visual representations are also looks similar. They can be differentiated by identifying the source and target elements that are connected to them. The compositional rules of each type ‘AssertedArtifactContext’ relationship are inherited the compositional rules of the ‘AssertedArtifactContext’ relationship.

To summarise, Figure 4.39 illustrates different types of ‘AssertedArtifactSupport’ relationship. Further explanation regarding the design rationale of each type ‘AssertedArtifactContext’ relationship can be found in Appendix A.7.

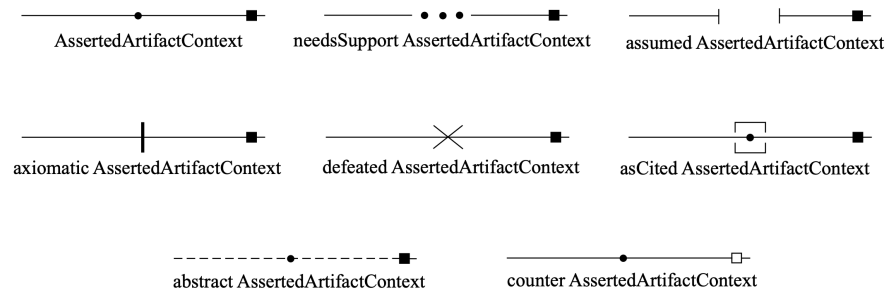


FIGURE 4.39: Types of ‘AssertedArtifactContext’ relationship

4.6.12 ‘ArgumentReasoning’

‘ArgumentReasoning’ is a sub-class of the ‘ArgumentAsset’ class. It is defined as a concrete class in the SACM specification document with the following semantics description:

"The ‘AssertedRelationship’ that relates one or more ‘Claims’ (premises) to another ‘Claim’ (conclusion), or evidence cited by an ‘ArtifactReference’ to a ‘Claim’, may not always be obvious. In such cases ‘ArgumentReasoning’ can be used to provide further description of the reasoning involved".

The ‘ArgumentReasoning’ has no sub-classes. It has inherited attributes from the ‘SACMElement’ class: ‘gid’, ‘isCitation’, and ‘isAbstract’. When the ‘isCitation’ or the ‘isAbstract’ attribute is applied to the ‘ArgumentReasoning’, in terms of visual representation creation, the design constraints of the ‘isCitation’ and the ‘isAbstract’ needs to be respected. For example, the visual representation of the ‘ArgumentReasoning’ needs to be drawn using dash line in order to respect the design constraint of ‘isAbstract’ attribute that makes the ‘ArgumentReasoning’ is declared to be part of an argument pattern or template. As for the ‘gid’ attribute, it is can be applied to the ‘ArgumentReasoning’ when the notation user consider it is necessary for an ‘ArgumentReasoning’ to have an element ID. Similar application of the used of element ID in an element that is similar with the ‘ArgumentReasoning’ in other assurance case notations can be found, for example, in the GSN Strategy.

Since the ‘ArgumentReasoning’ is a concrete class, its visual representation needs to be created. An ‘ArgumentReasoning’ is denoted using an annotation symbol as shown in Figure 4.40. The description can be written within the symbol, and a unique element identifier is placed at the top-left corner of the symbol when the ‘gid’ attribute is considered necessary to be applied by the users.

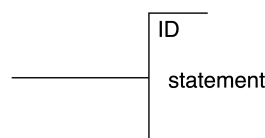


FIGURE 4.40: ‘ArgumentReasoning’ visual representation

An ‘ArgumentReasoning’ can be connected to ‘AssertedInference’ or ‘AssertedEvidence’ relationship in order to provides additional description or explanation of the asserted relationship. When used in a diagram, ‘ArgumentReasoning’ must be located on the left or right of the targeted element.

There is no specific rule regarding to the grammatical structure of ‘ArgumentReasoning’ statements. The ‘ArgumentReasoning’ statement shall contain a brief description of the argument approach.

The visual representation of the ‘ArgumentReasoning’ is not inherited the visual representation from any created SACM visual representations.

In creating the visual representation of the ‘ArgumentReasoning’, the design of the existing notation is also considered. The visual representation of the ‘ArgumentReasoning’ is inspired from the Flowchart annotation symbol and the BPMN text annotation symbol. In Flowchart, the annotation symbol is used to indicates additional information about a particular step in the program [87]. The annotation symbol is rendered as an open rectangle with a solid line connecting it to the corresponding symbol in the flowchart. Similar used of the text annotation symbol in BPMN where the Text Annotations in BPMN are used as a mechanism for a modeler to provide additional text information for the reader of a BPMN Diagram [88]. Since the role of the annotation symbol in flowchart and BPMN is similar with the role of the ‘ArgumentReasoning’, the adoption of the annotation symbol is considered applicable.

The purpose of the adoption of the Flowchart and BPMN annotation symbol to represents the SACM ‘ArgumentReasoning’ visual representation is to provides a clue, at least, to the user who have prior knowledge or experience with the BPMN or flowchart notation. A clash with existing notation, in this context, is considered avoided.

In terms of Semantic transparency, the annotation symbol is considered as a symbol-type visual representation for the notation user who have no prior experience related to used of the Flowchart notation because they need to learn in order to understand the meaning of it. However, if the user have previous experience in using Flowchart notation including the annotation symbol, the visual representation of the ‘ArgumentReasoning’ can be considered as index-type visual representation because they might recall the meaning of the symbol.

Further description related to the visual variables used and the other design rationale aspects that are considered in the creation of the ‘ArgumentReasoning’ can be read on Appendix A.8.

4.6.13 ‘metaClaim’

‘metaClaim’ is an association of the ‘Assertion’ class that associates the ‘Assertion’ class to the ‘Claim’ class. Semantically, the ‘metaClaim’ association is described as follow in the SACM specification document:

"*metaClaim*': 'Claim'[0..*] - references Claims concerning (i.e., about) the 'Assertion' (e.g., regarding the confidence in the 'Assertion')".

Based on the semantics, a 'metaClaim' can be used as a reference Claim that described about a particular 'Assertion' such as the 'Claim' element. The example of the 'metaClaim' application can be a relationship that connects a 'Claim' that arguing about a particular 'Claim', e.g., related to the confidence aspect of the 'Claim'.

Based on the literature related to assurance case theory and applications, one of the concrete example of the 'metaClaim' is related to the application of the *Assurance Claim Point* that are widely used in GSN.

The visual representation of the 'metaClaim' is created as illustrated in Figure 4.41.



FIGURE 4.41: 'metaClaim' visual representation

The visual representation of the 'metaClaim' is created by considering its semantics. It is created as a relationship that can be used to connects between a 'Claim' element and an 'Assertion' type such as another 'Claim' element.

A 'metaClaim' is rendered as a line with a crow's foot head-line type located at a line-end that is used as a pointer to indicates the target element of the 'metaClaim' relationship. The other line-end is drawn without any decoration of line-head that can be used to indicates the source element of the 'metaClaim' relationship.

When used in a diagram, a 'metaClaim' relationship starts from its source element which is must be a 'Claim' element and pointing to its target element that must be a type of an 'Assertion' such as 'Claim' element. The 'Claim' as the source element of the 'metaClaim' relationship can be used as a 'Claim' that arguing about the 'Assertion' that the 'metaClaim' relationship is pointing. In this case, the crow's foot is important to indicates the element that is being argued by the 'Claim' of the 'metaClaim' relationship.

The visual representation of the 'metaClaim', in terms of its line-head type, is different with the visual representation of the other type relationships such as the 'AssertedInference'.

The type of the line-head that is used in the 'metaClaim' visual representation is rendered using a 1-D type line-head because the 'metaClaim' is not inheriting the design constraints of the 'AssertedRelationship' (e.g., must be a 2-D type line-haed that can be used to declared the default state of the relationship and its counter purpose).

In term of Semantic transparency, the visual representation of the 'metaClaim' is considered as a symbol-type visual representation because the notation user needs to learn in order to understand

the meaning of it. The icon and index-type visual representation has been considered, however, the difficulty in finding such types of visual representation that can represent the semantics of ‘meta-Claim’ influenced the decision that the ‘metaClaim’ visual representation needs to be created based on symbol-type visual representation.

More description regarding the visual variables used and the other design constraints considered related to the creation of the ‘metaClaim’ visual representation can be read on Appendix A.9.

4.6.14 ‘ArgumentPackage’ and its variants

‘ArgumentPackage’ is a concrete sub-class of the ‘ArgumentationElement’ class. Its semantics is described as follows in the SACM specification document:

“ArgumentPackages’ contain structured arguments. These arguments are composed of ‘ArgumentAssets’. ‘ArgumentPackages’ elements can also be nested”.

Based on the semantics, ‘ArgumentPackage’ is the containing element for a structured argument represented using the SACM Argumentation Metamodel. It also can be seen as a packaging mechanism for representing assurance cases that contain structure arguments. The ‘ArgumentPackage’ can be used to support the concept of modularity in assurance case development.

Since the ‘ArgumentPackage’ is a concrete class and also part of the SACM argumentation metamodel, its visual notation needs to be created. Similar with the approach in creating the visual representation of the SACM elements, the creation of the ‘ArgumentPackage’ visual representation is considering its semantics, in terms of, prioritising the icon-type visual representation that can suggest the meaning of the element as described in its semantics. There is no inherited design constraints from its super-class other than the general design constraints that are adopted to develop the SACM notation. However, it is important to be noted that the creation of the ‘ArgumentPackage’ visual representation should also consider the application of it when being inherited and applied by its concrete sub-classes and other semantically related classes in SACM metamodel (i.e., ‘AssurancePackage’, ‘ArtifactPackage’, and ‘TerminologyPackage’) for the purpose of the visual representation creation.

The package icon is used to visually represent the concept of Package in general in SACM. It is inspired by the convention of a container icon (see Figure 4.42). In this case, the package icon in SACM is designed with three main compartments based on a specific purpose:

- The largest compartment is prepared for the text as the description of the ‘ArgumentPackage’.
- The top-right compartment is prepared to place the general type of Package defined in SACM metamodel:
 - ‘AssurancePackage’: no decoration needed to indicate the Assurance Package.

- ‘ArgumentPackage’: indicated using a rectangle.
 - ‘ArtifactPackage’: indicated using the ‘Artifact’ visual representation.
 - ‘TerminologyPackage’: indicated using "T" (write in Bold) to denotes "Terminology".
- Since the Package in SACM can be specified into two: Binding Package and Interface Package, the bottom-right compartment is prepared to place the specific type of Package:
 - Binding Package: indicated using (overlap) double-circles decoration to denotes that the package is used bind the package elements.
 - Interface Package: indicated using a-circle-head line decoration to represents the meaning of an Interface Package.

Figure 4.42 illustrates the convention used to create and identify each type of Package in SACM notation. To identify and read the type of SACM Package we need to observe the general type of the Package first, it can be either an ‘AssurancePackage’, ‘ArgumentPackage’, ‘ArtifactPackage’, or ‘TerminologyPackage’. After that, we need to observe the specific type of the Package, it can be either an Interface or Binding Package.

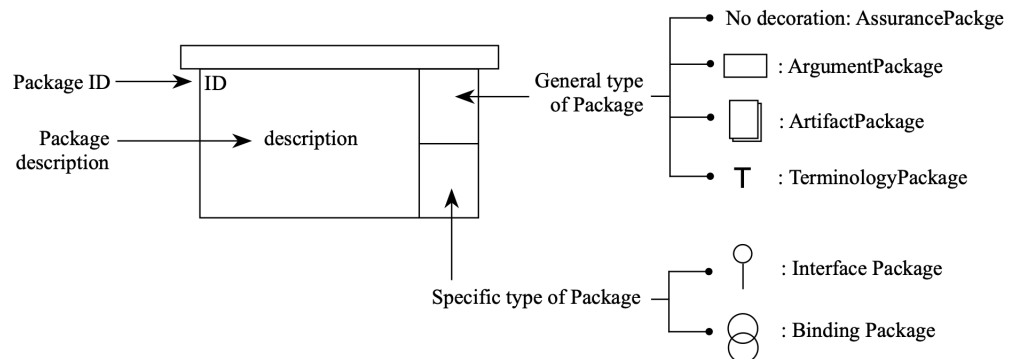


FIGURE 4.42: SACM Package convention

If the notation user considered it is necessary for a Package to have an element ID, it can be placed on the top-left corner within the description compartment of the visual representation.

In the case of the ‘ArgumentPackage’, its created visual representation is illustrated in Figure 4.43.

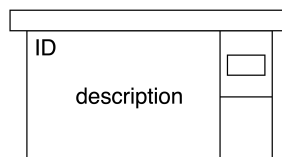


FIGURE 4.43: ‘ArgumentPackage’ visual representation

The ‘ArgumentPackage’ has two concrete sub-classes: ‘ArgumentPackageBinding’ and ‘ArgumentPackageInterface’. The visual representation of these sub-classes are created based on visual inheritance respecting the inherited design constraints of the ‘ArgumentPackage’ and following the convention rules as illustrated in Figure 4.42.

Figure 4.44 illustrates the types of the ‘ArgumentPackage’ in SACM along with the relationship that can be used to associates them. The visual representation of the ‘ArgumentPackage’, ‘ArgumentPackageBinding’, and ‘ArgumentPacakgeInterface’ looks similar to each other since they have similar semantics. A specific design identity, such as a circle with a vertical line for the ‘ArgumentPacakgeInterface’, and double-circles that overlap each other for the ‘ArgumentPackageBinding’, is added to differentiate them in order to denote their specific semantics.

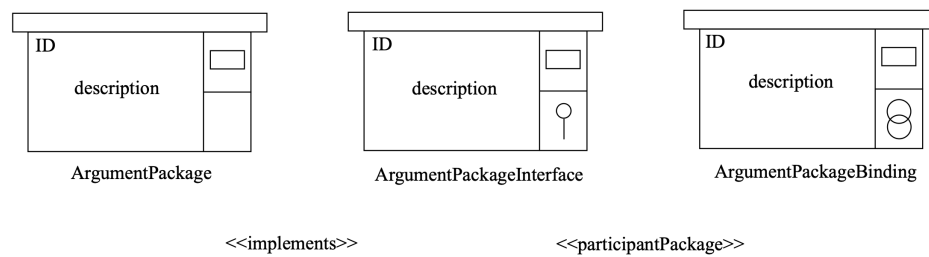


FIGURE 4.44: Types of ‘ArgumentPackage’

When used in a diagram, an ‘ArgumentPackage’ can be associated to ‘ArgumentPackagesInterface’ via the ‘implements’ relationship. An ‘ArgumentPackage’ can be nested if and only if the the contained element is also the ‘ArgumentPackages’.

‘ArgumentPackageInterface’ is an ‘ArgumentPackage’ type that can be used as an interface to declare one or more argumentation elements that is contained in an ‘ArgumentPackage’. An ‘ArgumentPackage’ may declare one or more ‘ArgumentPackageInterface’.

An ‘implements’ relationship is used to associates between ‘ArgumentPackage’ and ‘ArgumentPackageInterface’ element that means one or more argumentation elements that are defined in an ‘ArgumentPackage’ is declared to be part of an ‘ArgumentPackageInterface’. These elements then must be a type of citation element that cites the elements that are located in the ‘ArgumentPackage’.

An ‘implements’ relationship is rendered as a solid (texture) line where an «implements» keyword is located around the middle of the line to indicates that it is an ‘implements’ relationship. An ‘implements’ relationship has no decoration in terms of the line-head in both line-end since it is not inherited any design constraints that requires the ‘implements’ relationship to show its source and target elements.

‘ArgumentPackageBinding’ can be used to bind the participant packages by means of argument elements that connect the cited elements of the participant packages. In other words, ‘ArgumentPackageBinding’ can be used to record the argument that connects the arguments of two or more contained in

the ‘ArgumentPackages’.

To associate an ‘ArgumentPackageInterface’ and ‘ArgumentPackageBinding’, the notation user can use a ‘participantPackage’ relationship. A ‘participantPackage’ relationship is visually represented as a solid (texture) line where a «participantPackage» keyword is written around the middle of the line. There is no line-head type decoration to be added as part of the visual representation of the ‘participantPackage’ relationship since it is not inherited any design constraint that requires the ‘participantPackage’ relationship need to show or indicates its source or target elements.

The visual representation of the ‘participantPackage’ is designed to be similar with ‘implements’ relationship visual representation due to consider graphic complexity design principle (i.e. managing the number of the visual representations of a notation). They can be differentiated by identifying the keyword that attach to the line and the elements that are connected to the relationship.

The ‘ArgumentPackage’ class also has a composition relationship, that is described as follow in the SACM specification document:

"'argumentationElement': 'ArgumentationElement'[0..] (composition) – a collection of 'ArgumentationElements' forming a structured argument".*

The composition relationship of the ‘ArgumentPackage’ that associates the ‘ArgumentPackage’ to ‘ArgumentationElement’ class helps to define the rules in using the ‘ArgumentPackage’ that can be summarised as follow:

- An ‘ArgumentPackage’ can be used to contain the ‘ArgumentationElements’ such as ‘Claims’, ‘ArtifactReferences’, ‘ArgumentReasoning’, and ‘AssertedRelationship’ types forming a structured argument.
- An ‘ArgumentPackage’ can not be created or declared without any ‘ArgumentationElement’ contained in the ‘ArgumentPackage’.

The ‘ArgumentPackage’ class has inherited attributes from the ‘SACMElement’ class: ‘gid’, ‘is-Abstract’, and ‘isCitation’. The application of the ‘gid’ attribute to the ‘ArgumentPackage’ visual representation has been discussed previously in this sub-section. As for the ‘isAbstract’ and ‘isCitation’ attribute, in terms of the visual representation, theoretically, it is possible to be applied to the the ‘ArgumentPackage’ visual representation. The creation of the ‘ArgumentPackage’ visual representation that applied the ‘isAbstract’ and ‘isCitation’ attribute must comply to the design constraints of each attribute, .e.g, the ‘ArgumentPackage’ visual representation must be drawn using dash-line in order to define the ‘ArgumentPackage’ as an ‘abstract ArgumentPackage’. However, the related practical examples of ‘ArgumentPackage’ that is declared as an ‘abstract ArgumentPackage’ and ‘as-Cited ArgumentPackage’ are hardly to be found in the literature. Therefore, the application of the

'isAbstract' and 'isCitation' in the 'ArgumentPackage' class is only theoretically applicable and still requires further research such as related to its application in real cases.

The visual representation identity of each type of 'ArgumentPackage' and its relationship can be read further in Appendix A.10.

4.6.15 'ArgumentGroup'

'ArgumentGroup' is a concrete sub-class of the 'ArgumentationElement' class. Its semantics is described as follow according to the SACM specification document:

"'ArgumentGroup' can be used to associate a number of 'ArgumentElements' to a common group (e.g. representing a common type or purpose, or being of interest to a particular stakeholder). The name and the description of the 'ArgumentGroup' should provide the semantic for understanding the 'ArgumentGroup'. 'ArgumentGroups' serve no structural purpose in the formation of the argument network, nor are they meant as a structural packaging mechanism (this should be done using 'ArgumentPackages')".

Based on the semantics, 'ArgumentGroup' can be used to associate a number of argumentation elements of SACM to a common group, for example, representing a common type of elements or purpose or being of interest to a particular user.

Since the 'ArgumentGroup' is a concrete class and part of the SACM argumentation metamodel, its visual representation need to be created. The created visual representation of the 'ArgumentGroup' is illustrated in Figure 4.45.

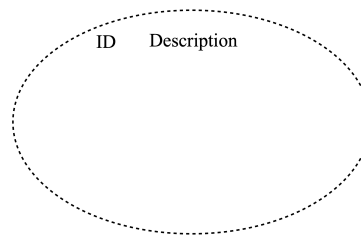


FIGURE 4.45: The 'ArgumentGroup' visual representation

An 'ArgumentGroup' relationship is visually represented using dotted-line that can be used to group related elements of SACM where the description of the 'ArgumentGroup' can be written within the dotted-lines at the top-part of the visual representation along with an element ID, in the case the notation user considered it is necessary for the 'ArgumentGroup' to have an ID. The elements that are considered to be part of the same group must be located within the dotted-lines.

According to the SACM specification document, 'ArgumentGroup' has a association relationship named 'argumentationElement' that connects the 'ArgumentGroup' class to the 'ArgumentationElement' class that is be described as follow:

"'argumentationElement': 'ArgumentationElement'[0..] – an optional collection of 'ArgumentationElements' organised within the 'ArgumentGroup'".*

The 'argumentationElement' association can be used to define the application of the 'ArgumentGroup' when used in a diagram: it can be used to associates a number of 'ArgumentationElement' (i.e. SACM argumentation elements) to a common group.

An 'ArgumentGroup' also can be seen as a spatial relationship, it serve no structural purpose in the formation of the argument network, nor are they meant as a structural packaging mechanism (this should be done using 'ArgumentPackages'). There is no specific restriction in drawing the visual representation of the 'ArgumentGroup' other than the type of the line in drawing the 'ArgumentGroup' (i.e. using dotted lines). There is also no specific geometrical shape form is defined in creating the 'ArgumentGroup' visual representation.

The visual representation of the 'ArgumentGroup' is inspired from the visual representation of elements grouping mechanisms in I* notation. We have considered other visual representations to represent 'ArgumentGroup' in SACM. For example, in the SACM specification document [9], the SACM metamodel as a complete specification is drawn by utilising Colour to differentiate each areas of SACM component. For example, the Argumentation metamodel area is highlighted using the green colour. Such mechanism has been considered to be used to represents the 'ArgumentGroup'. However, since we decided not using the visual variable Colour in our design process, therefore, grouping related elements using Colour is not being adopted. Alternatively, we adopted the I* approach for grouping several related elements using dotted-lines. More description related the visual variables used and the other design constraints considered in the creation of the 'ArgumentGroup' can be read in Appendix A.11.

The 'ArgumentGroup' has no sub-classes. It has inherited attributes from the 'SACMElement' class: 'gid', 'isAbstract', and 'isCitation'. The application of the 'gid' attribute has been described previously as part of the creation of the 'ArgumentGroup' visual representation in this sub-section.

Theoretically, the 'isAbstract' and 'isCitation' attribute can be applied in the 'ArgumentGroup' class since it is inherited to the 'ArgumentGroup' class. However, the practical examples of the 'abstract ArgumentGroup' and 'asCited ArgumentGroup' are hardly to be found in the literature. Therefore, the application of the 'isAbstract' and 'isCitation' in the 'ArgumentGroup' class currently are only theoretically applicable. Further research is still needed, especially, related to its application in real cases.

4.7 Design rationale of the created visual representation

There are 33 visual representation identity tables created for the SACM notation. For example, the visual representation identity of 'Claim' and its variants, 'AssertedInference' and its variants, and

‘ArtifactReference’ element. The created visual representation identity of each visual representation can be seen as the design rationale behind the creation of the visual representation that also can be used for further development of the notation.

Considering the number of visual representation identity tables that needs to be presented in this section, they can be read further in Appendix A - Visual Representation Identity of SACM Notation.

4.8 Summary of SACM notation

To summarise, Figure 4.46 presents the overall of SACM argumentation notation that has been discussed in this chapter.

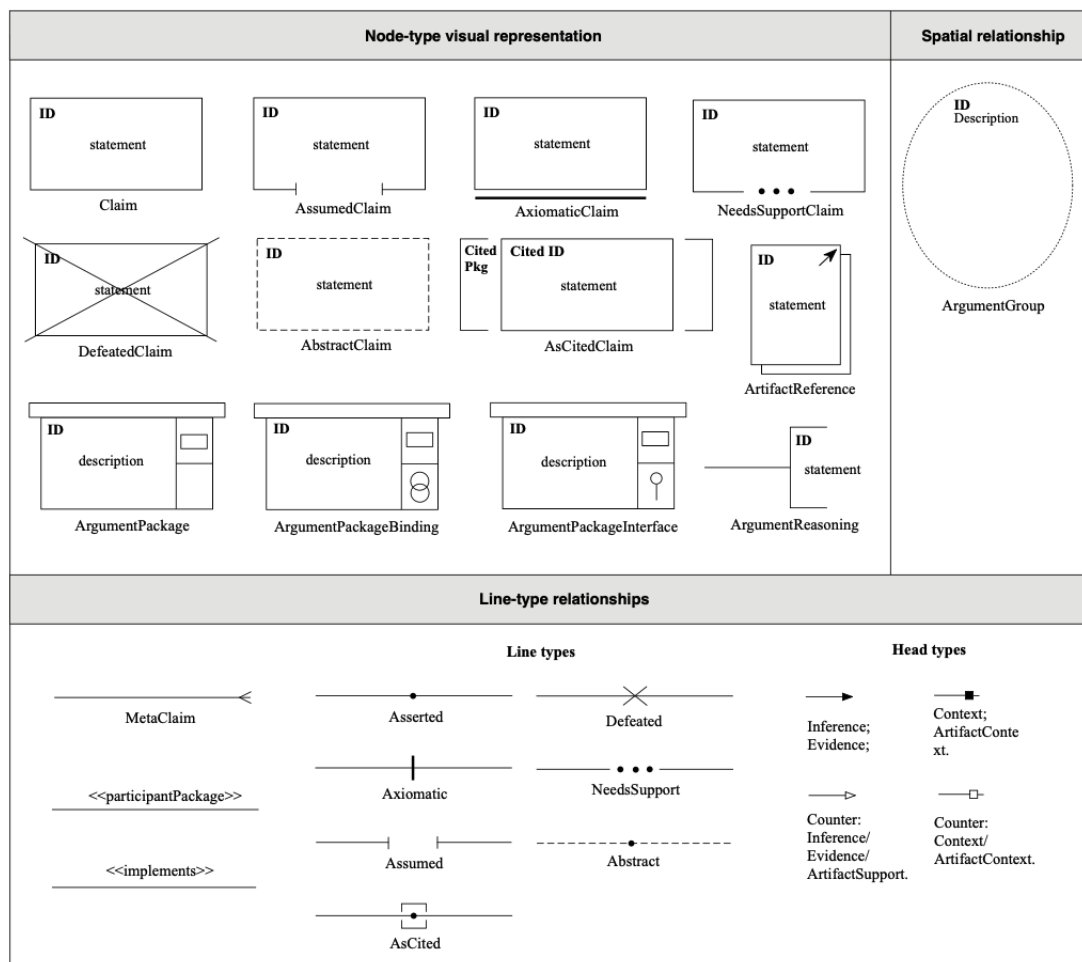


FIGURE 4.46: Summary of SACM argumentation notation

In Figure 4.46, the ‘Claim’ and its variants, ‘ArgumentPackages’, ‘ArtifactReference’, and ‘ArgumentReasoning’ can be seen as the node-type visual representation. The ‘AssertedRelationship’ types,

‘metaClaim’, ‘participantPackage’, and ‘implements’ are categorised as the line-type visual representation. Finally, we considered the ‘ArgumentGroup’ as the spatial relationship visual representation in SACM notation.

4.9 Application examples

In this section, the application examples of the created visual representations of the SACM argumentation elements are presented. To show the use of the SACM elements in a graphical diagram, an assurance case for a deep learning system used for retinal disease diagnosis and referral is adopted and modified from [89].

Figure 4.47 illustrates the example of ‘Claims’ supporting another ‘Claim’ where the ‘AssertedInference’ relationship is used to associate these ‘Claims’. The "ML Assurance Claim" is the main ‘Claim’. It is supported by "Segmentation Network" and "Classification Network" ‘Claims’. These ‘Claims’ are connected via ‘AssertedInference’ relationship that is drawn from the supporting ‘Claims’ to the supported ‘Claim’.

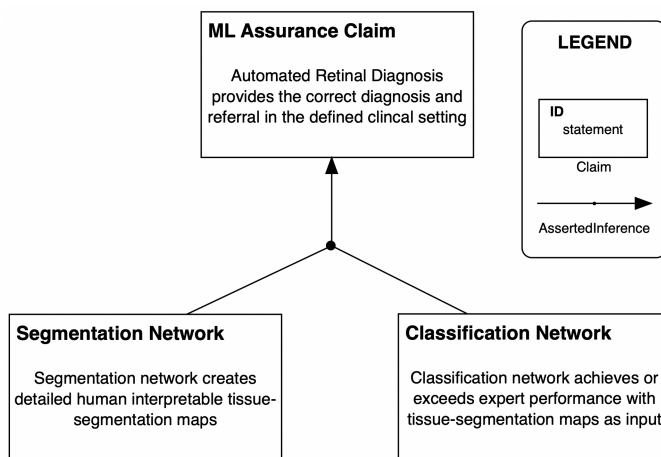


FIGURE 4.47: Example of ‘Claims’ supporting another ‘Claim’. The ‘AssertedInference’ relationship is used to associates these elements.

A ‘Claim’ can be declared in a particular scope or context. This can be done by asserting an ‘ArtifactReference’ as a reference to a contextual information that must be drawn and located horizontally relative to the ‘Claim’, where an ‘AssertedContext’ relationship can be used to declare the relationship between the ‘ArtifactReference’ (as a context) and the ‘Claim’.

Figure 4.48 illustrates the use of ‘ArtifactReferences’ that provides the context for the interpretation and scoping of a ‘Claim’. In this case, the "Clinical Setting" and "Automated Retinal Diagnosis" ‘ArtifactReferences’ are used as the elements that provides context for the interpretation and scoping of the "ML Assurance Claim" ("Clinical Setting" and "Automated Retinal Diagnosis" refer to a specific artifact that can helps to scope the interpretation of the ‘Claim’). The ‘AssertedContext’ relationship is used to associates ‘ArtifactReferences’ (as a context) and ‘Claim’.

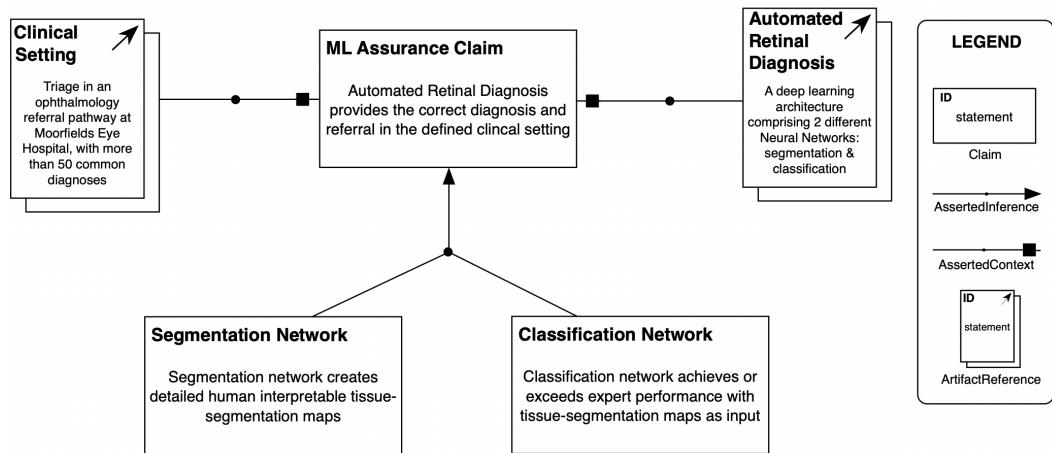


FIGURE 4.48: Example of 'ArtifactReference' providing context to a 'Claim'. The 'AssertedContext' relationship is used to associate these elements.

In SACM, it is possible to provide context to the (asserted) relationship, for example, that connect between 'Claims'. Figure 4.49 illustrates the example of 'ArtifactReferences' that provides context to the 'AssertedInference' relationship.

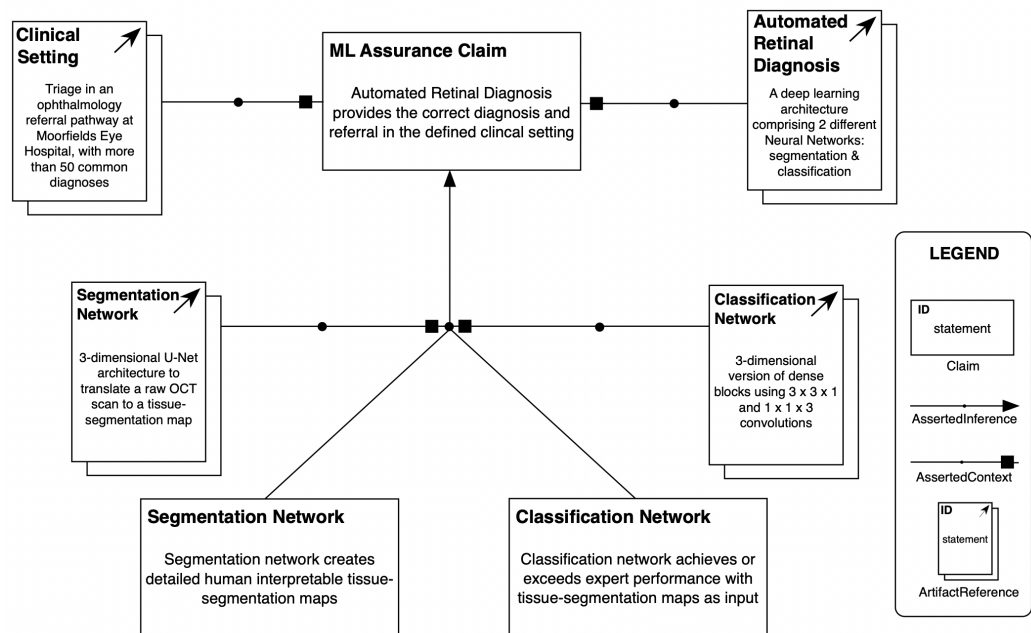


FIGURE 4.49: Example of 'ArtifactReference' providing context to the 'AssertedInference' relationship. The 'AssertedContext' relationship is used to associate these elements.

An 'ArtifactReference' that is used as a reference to a contextual information can be asserted and drawn horizontally relative to the relationship that it is targeted. The relationship used to associate the 'ArtifactReference' and the targeted relationship is declared using 'AssertedContext' relationship.

As presented in Figure 4.49, the "Segmentation Network" and the "Classification Network" 'ArtifactReferences' are used to provide the context for the 'AssertedInference' that is declared to associate the "ML Assurance Claim" and its supporting 'Claims' ("Segmentation Network" and "Classification Network"). The 'AssertedContext' relationship is used to associate the 'ArtifactReferences' describing the networks with the 'AssertedInference' relationship.

An 'ArgumentReasoning' can be added to the assurance case diagram to provide additional description or explanation of the asserted relationship. As illustrated in Figure 4.50, 'ArgumentReasoning' "Assurance Case Strategy" is added to the diagram in order to provide explanation about the assertion between 'Claims'.

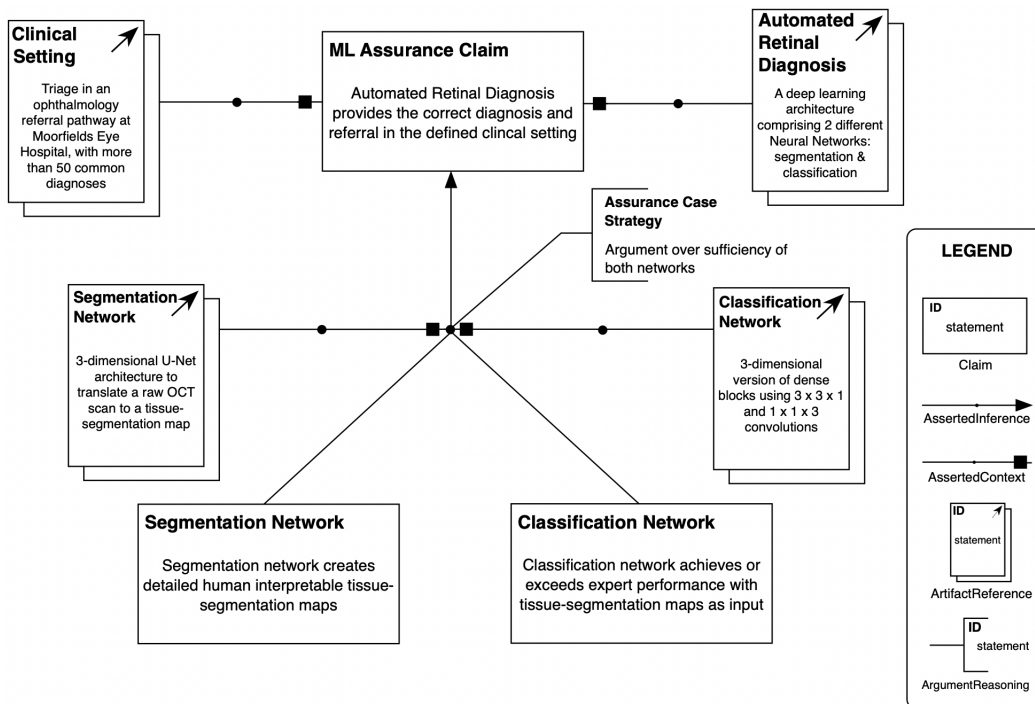


FIGURE 4.50: Example of 'ArgumentReasoning' providing additional description to the assurance case.

In some cases, other than using an 'ArtifactReference' to provide a context to a particular 'Claim' or 'AssertedInference', a 'Claim' can also be asserted as necessary context (i.e. a precondition) for another 'Assertion' such as a 'Claim' or 'AssertedInference'.

Figure 4.51 shows the example of a 'Claim' that provides necessary context for another 'Claim', in this case, the "Gold Standard" as an 'assumed Claim' provides the context for the "Classification Network" 'Claim'. In contrast to the previous examples, the 'assumed claim' is providing additional information as context, rather than referencing to an artefact that provides the context. The relationship between these elements is declared using an 'AssertedContext' relationship.

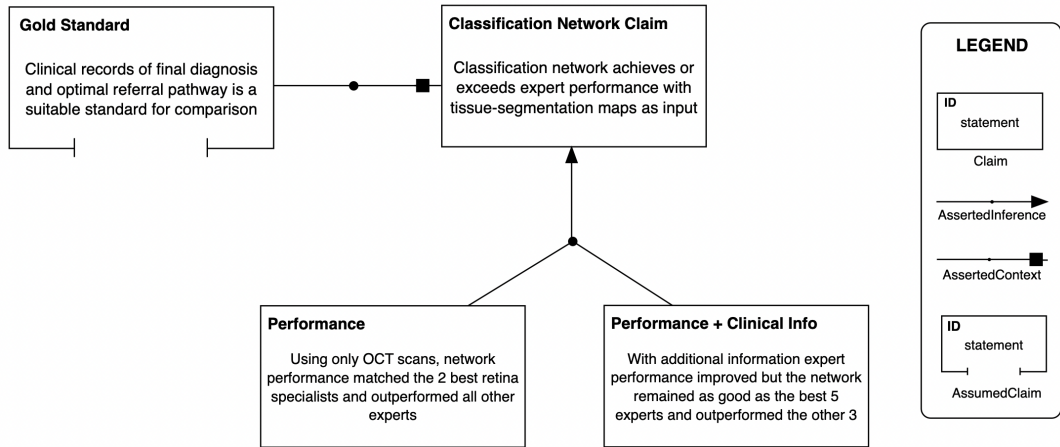


FIGURE 4.51: Example of a 'Claim' provides necessary context for another 'Claim'. The 'AssortedContext' relationship is used to associates these elements.

An 'ArtifactReference' can also be used as a reference to evidential information to support one or more 'Claims'. The example of this case can be seen in Figure 4.52.

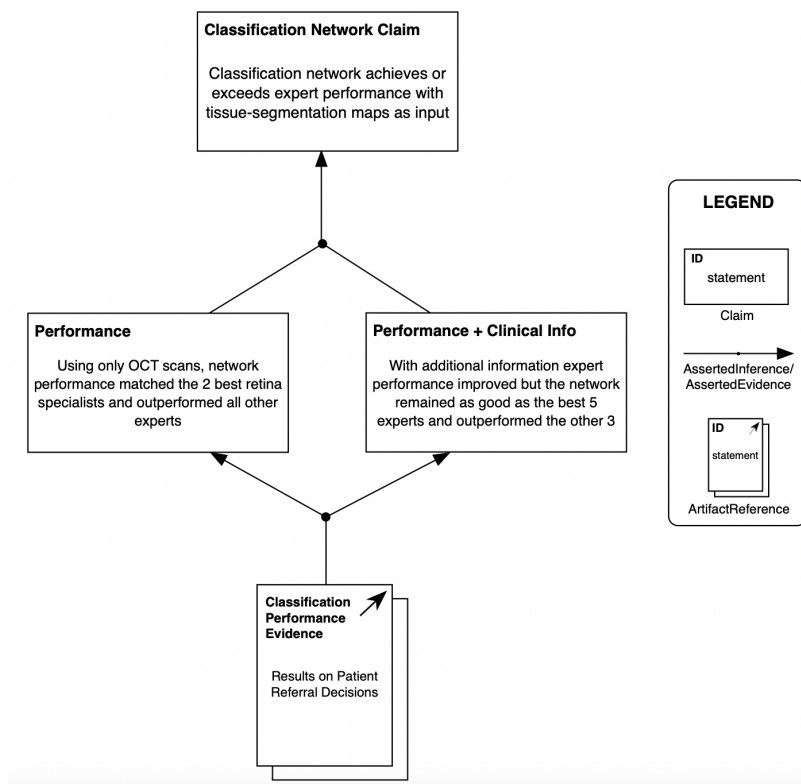


FIGURE 4.52: Example of an 'ArtifactReference' as evidence to support 'Claims'. The 'AssortedEvidence' relationship is used to associates these elements.

An 'ArtifactReference' that is used as a reference to evidential information to support one or more 'Claims' must be located below the supported 'Claim'. The relationship between them must be declared using an 'AssortedEvidence' relationship. Both the position, as well as the relationship type indicate whether an 'ArtifactReference' is being used as context or evidence in the argument.

The example in Figure 4.52 illustrates an ‘ArtifactReference’ as evidence ("Classification Performance Evidence") to support the "Performance" and the "Performance + Clinical" ‘Claims’. The ‘AssertedEvidence’ relationship is used to associate these elements.

In the case where it is necessary for an ‘ArtifactReference’ (as evidence) to be specified in a particular context, another ‘ArtifactReference’ (as context) can be asserted and drawn horizontally relative to the ‘ArtifactReference’ (as evidence) as the target element. The relationship between these ‘ArtifactReferences’ can be declared using the ‘AssertedArtifactContext’ relationship. The ‘AssertedArtifactContext’ starts from the ‘ArtifactReference’ (as context) pointing to the ‘ArtifactReference’ (as evidence). This indicates that the ‘ArtifactReference’ (as context) provides a necessary context for the ‘ArtifactReference’ (as evidence).

Similarly, in the case where an ‘ArtifactReference’ (as context) needs further support from another ‘ArtifactReference’ as evidence, this can be done by asserting an ‘ArtifactReference’ (as evidence) that is placed below the ‘ArtifactReference’ (as context). The relationship between these elements can be declared using an ‘AssertedArtifactSupport’ relationship that starts from the ‘ArtifactReference’ (as evidence) and points to the ‘ArtifactReference’ (as context).

In Figure 4.53 we shows another example of the ‘Claim’ variant, in this case is the ‘axiomatic Claim’. We modified the assurance case diagram as illustrated in Figure 4.50 in order to show the use of an ‘axiomatic Claim’ that is presented in Figure 4.53.

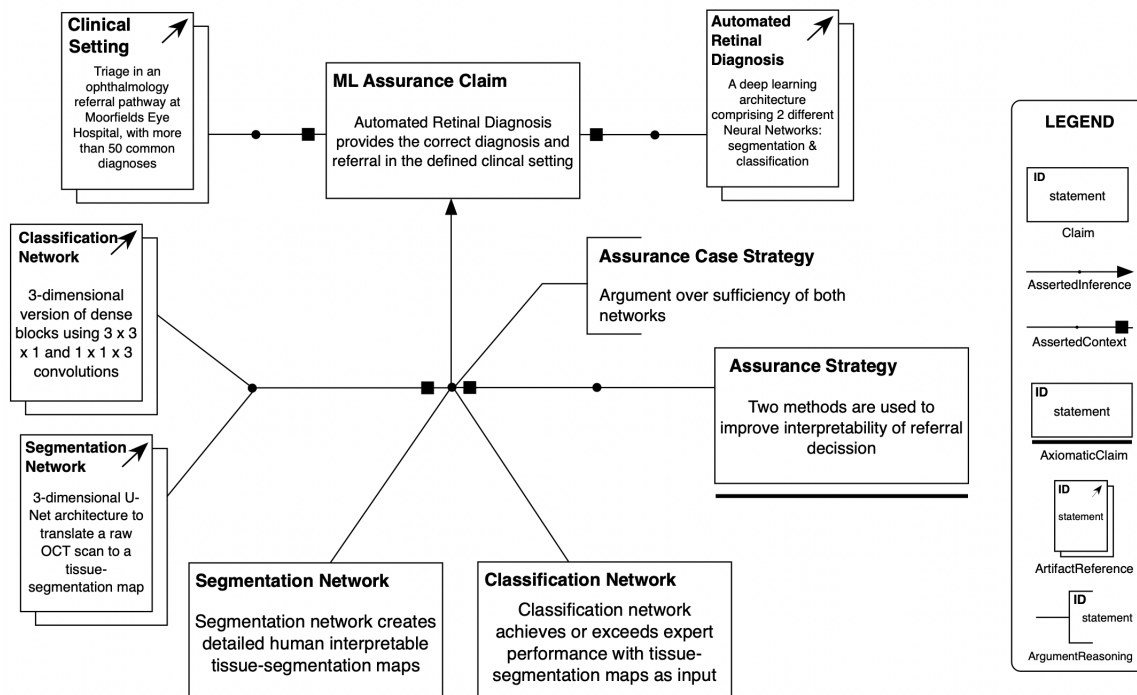


FIGURE 4.53: Example of an ‘axiomatic Claim’

We added an ‘axiomatic Claim’ called "Assurance Strategy" (in Figure 4.53) to justify the ‘Asserted-Inference’ relationship that associates the ‘Segmentation Network Claim’ and the ‘Classification Network Claim’ to the supported claim "ML Assurance Claim". The relationship between the "Assurance Strategy" ‘axiomatic Claim’ and the ‘AssertedInference’ relationship is declared using ‘AssertedContext’ relationship.

Another example of the application of the ‘Claim’ variant is shown in Figure 4.54. The "Segmentation Outcome Interoperability" ‘Claim’ is declared as an ‘asCited Claim’ that means it is citing another ‘Claim’ ("Segmentation Map Result" ‘Claim’) that is defined and developed in a different argument package (in this case an ‘ArgumentPackage’ named "Arg.Pkg.1") that contains the argument regarding human interoperability of segmentation maps.

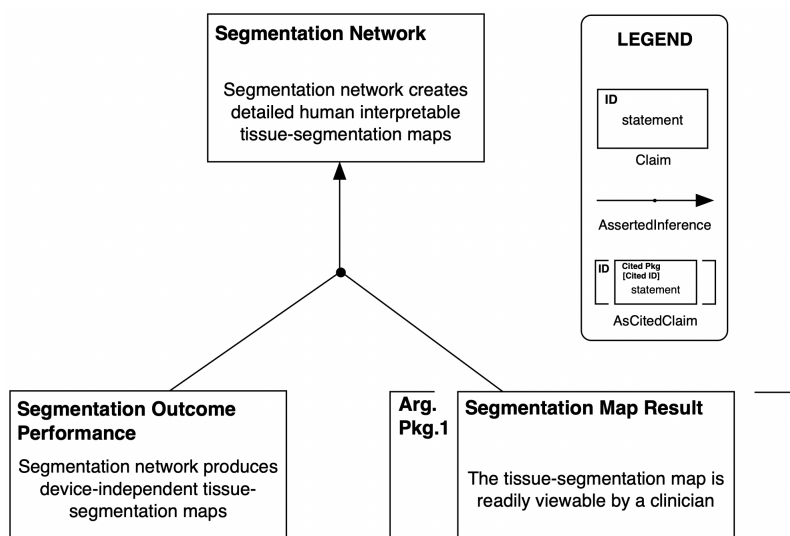


FIGURE 4.54: Example of an ‘asCited Claim’

As for the ‘needsSupport Claim’, the example of it can be seen in Figure 4.55. The "Ambiguous Regions" ‘Claim’, that supports the "Segmentation Outcome Performance" ‘Claim’, is defined as a ‘needsSupport Claim’. This means that this claim has not yet been developed and will require further argumentation and evidence to be provided.

Figure 4.56 shows an example dialectical assurance argument in SACM where a ‘Claim’ is declared as a ‘defeated Claim’ due to some counter evidence (via ‘ArtifactReference’ and ‘counter AssertedEvidence’).

For purpose of illustration, we modify and reconstruct the assurance argument fragment in Figure 4.55. The "Unambiguous Regions" ‘Claim’ in Figure 4.56 is declared as a ‘defeated Claim’ because it is countered by a counter-evidence of an inconsistent segmentation network result, which is cited via ‘ArtifactReference’. As a result, the tissue segmentation map can no longer be said to be consistent with the manually-generated one. To associate the counter-evidence (cited visa ‘ArtifactReference’) and the targeted claim, in this example, the ‘counter AssertedEvidence’ relationship is used.

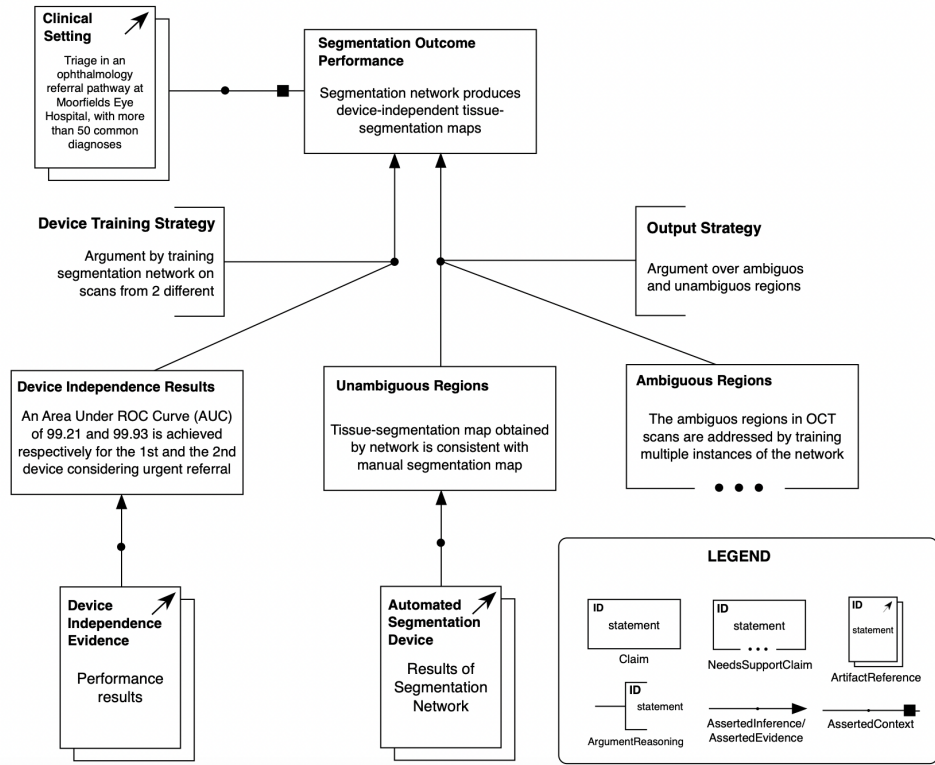


FIGURE 4.55: Example of a ‘needsSupport Claim’

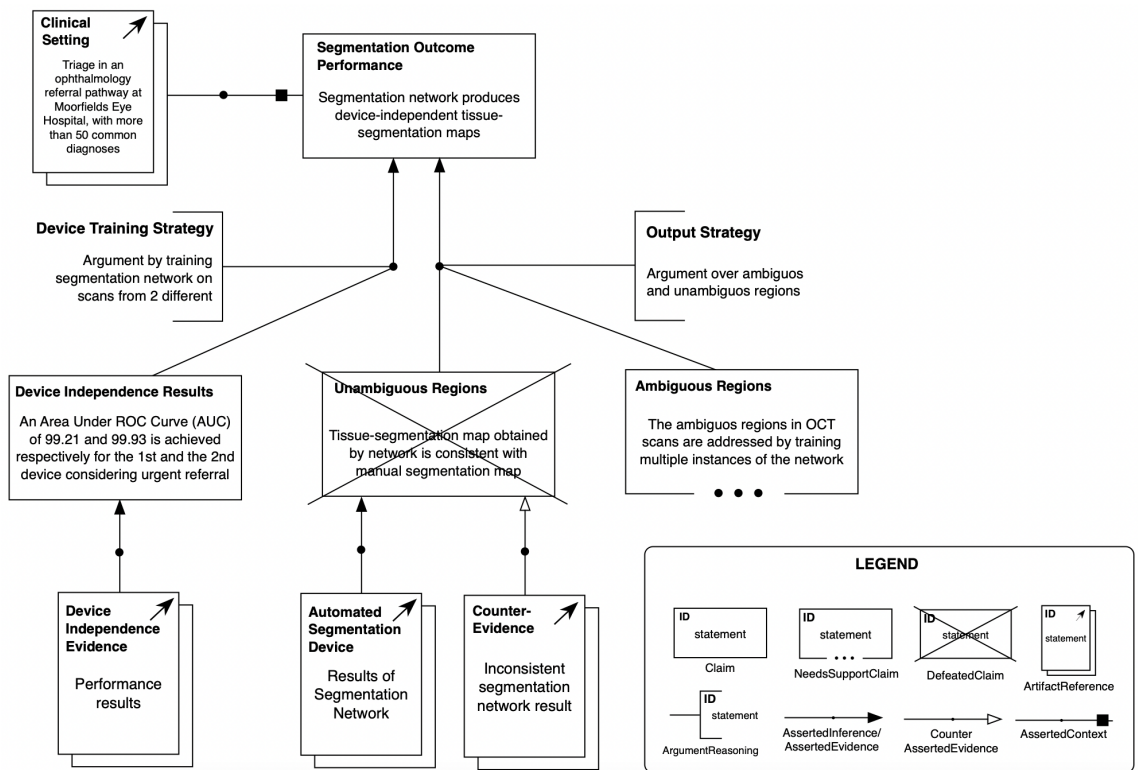


FIGURE 4.56: Example of dialectical assurance argument

Figure 4.57 shows the example of the abstract elements. In Figure 4.57, 'Claim' "Network" is defined as an 'abstract Claim'; there is some aspect of it that is generalised and requires instantiation. In this case the claim concerns a neural network achieving a defined standard. Such a claim could be made for many different neural networks, the curled braces indicate that the name of the specific network can be substituted. Similarly, the 'abstract ArtifactReferences' that provide context for the 'Claim' can be also instantiated with information relevant to the specific network considered in a particular case (such as the specific gold standard used or the particular training data set).

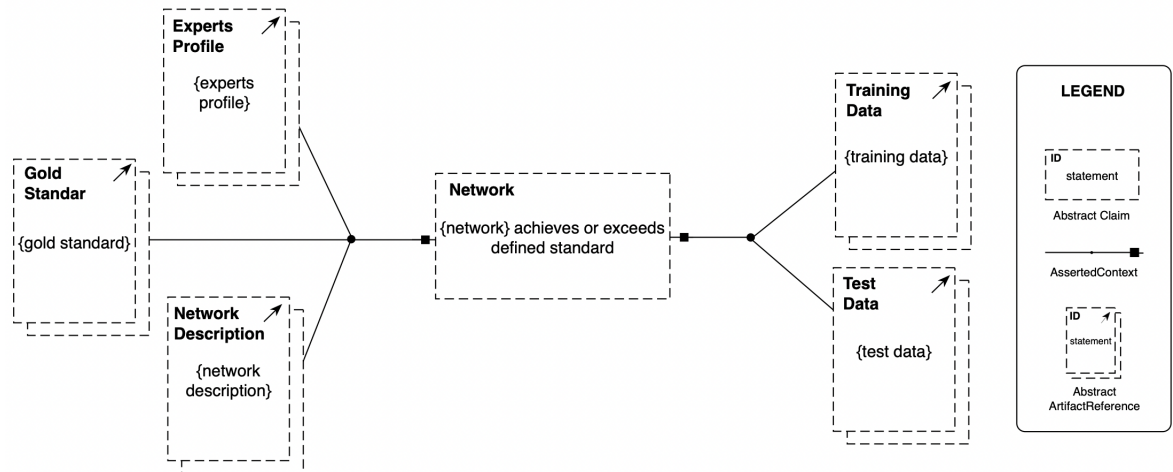


FIGURE 4.57: Example of abstract elements in SACM notation

Another example of the abstract elements can be seen in Figure 4.58. In this example, we modified and reconstruct an assurance case pattern/template fragment from [90] in order to show the application of the abstract elements including abstract relationship such as **optional** 'abstract AssertedContext', **multiple** 'abstract AssertedInference', and **choice** 'abstract AssertedInference'.

In Figure 4.58, "Goal: sw contribution" is declared as an 'abstract Claim'. It is supported by an 'abstract Claim' "Goal: SRAdd". The relationship between these 'Claims' is declared using a *multiple* 'abstract AssertedInference' which means the "Goal: sw contribution" 'Claim' can be supported by mutiple instantiations of the "Goal: SRAdd" 'Claim' in the instantiated assurance case model.

An 'axiomatic Claim' "Just: SRidentify" is asserted to support the 'abstract AssertedRelationship' between 'Claims' "Goal: sw contribution" and "Goal: SRAdd". The relationship between the "Just: SRidentify" and the 'abstract AssertedRelationship' is declared using *optional* 'abstract Asserted-Context' which indicates that "Just: SRidentify" is only required in certain circumstances; it must be decided at the time of instantiating the argument whether it is required or not.

As for the "Goal: SRAdd", it is supported by "Goal:SRSat" and "Goal: SRaddDes"; both of this 'Claims' are declared as a 'needsSupport abstractClaim'. The relationship between the supporting 'Claims' and the supported 'Claim' is declared using a *choice* 'abstract AssertedInference' relationship. This means that both "Goal: SRAdd" and "Goal: SRaddDes" can be supported by at least one

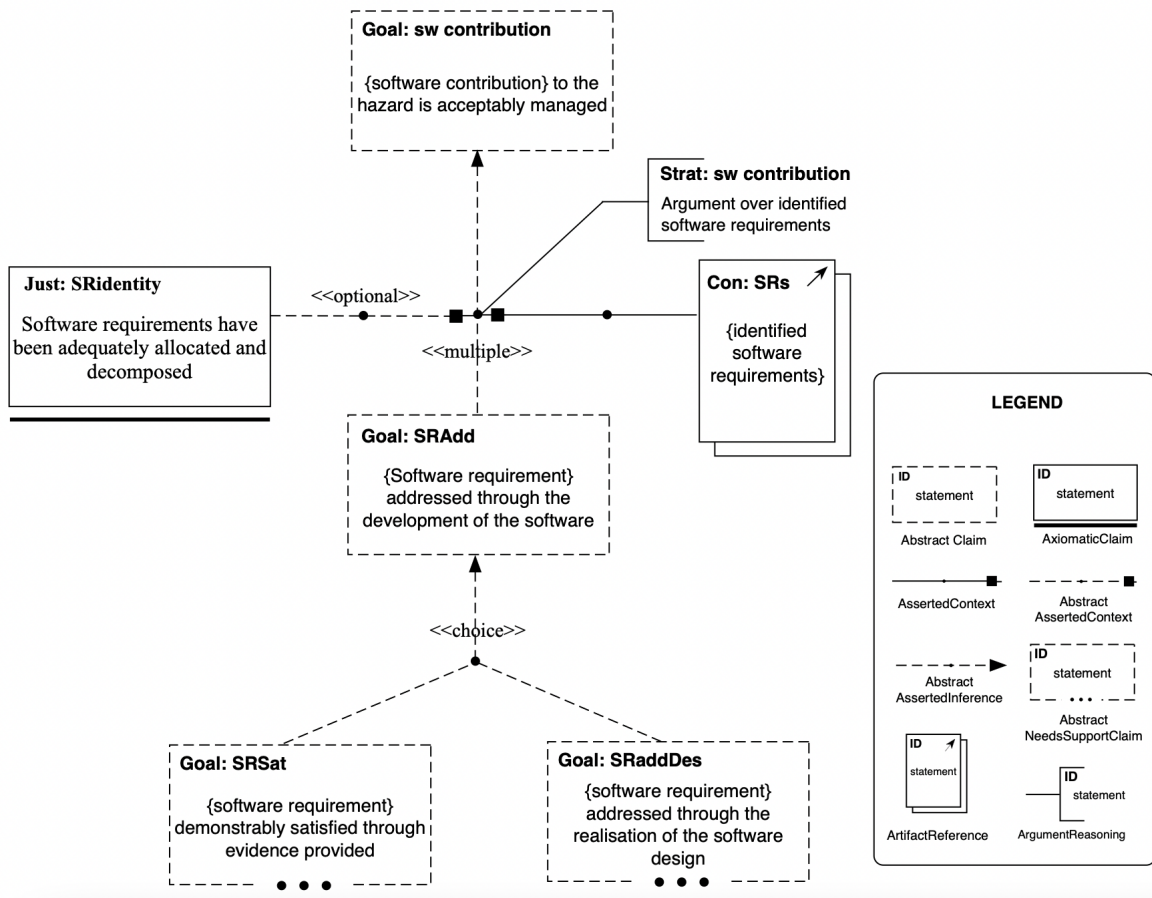


FIGURE 4.58: Another example of abstract elements in SACM notation

of the supporting claims; a choice must be made when instantiating the argument as to which of the claims is required for the particular system being considered (in this case, either the instantiation of the "Goal: SRAdd" or the "Goal: SRaddDes").

Figure 4.59 illustrates the example of the ‘metaClaim’. This example shows a continuation of the assurance case presented previously in Figure 4.47.

A ‘metaClaim’ is a relationship that can be used to associates a particular ‘Claim’ to a particular ‘Assertion’ (e.g. ‘Claim). In this context, the ‘Claim’ is used as a reference Claim that described about a particular ‘Assertion’, e.g., related to the confidence aspect of the ‘Claim’.

The "Classification Network Claim" in Figure 4.59 is supported by "Performance" and "Performance + Clinical" ‘Claim’. It is also scoped by the "Training Data", "Test Data", and "Validation Data" ‘ArtifactReferences’. A ‘Claim’, "MC1-ClassNet", is added to the diagram to argue about the confidence of the "Classification Network Claim". The relationship between the "MC1-ClassNet" ‘Claim’ and the "Classification Network Claim" is declared using a ‘metaClaim’ relationship (visualised using a crow’s foot line). This indicates that the "MC1-ClassNet" ‘Claim’ is used to presents an argument about the "Classification Network Claim".

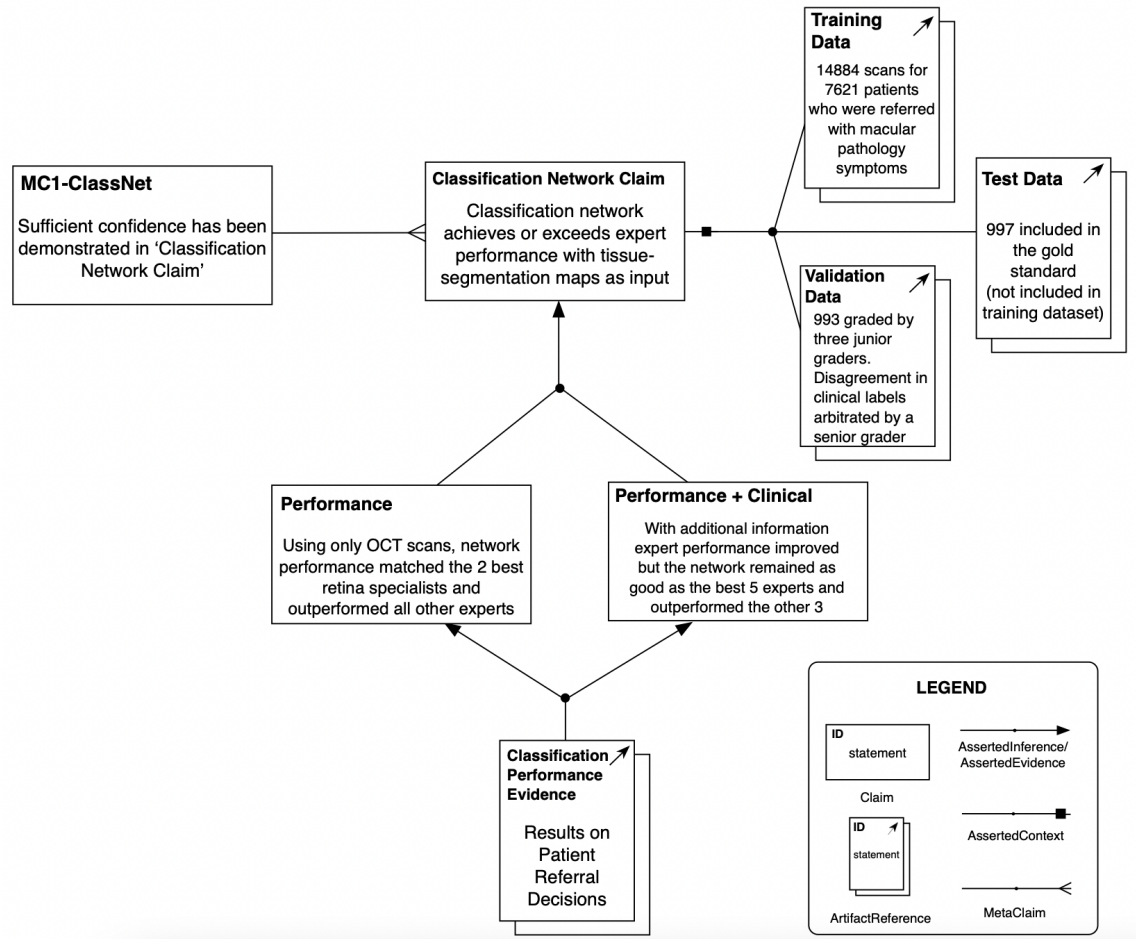


FIGURE 4.59: Example of 'metaClaim'

As for the 'ArgumentPackages', we provide an example adopted from [14] related to an engineering case study in European Train Control System (ETCS) in DEIS project to illustrate how 'Argument-Packages' are used in an assurance case. The scenario that are considered in this example is about the assurance cases of on-board and side-track components of ETCS that are integrated to form an overall assurance case. This example is modified for simplicity and reconstruct to be presented using SACM notation as illustrated in Figure 4.60.

"OB.API", in Figure 4.60, is an 'ArgumentPackage' contains the argument regarding the safety of the on-board component of ETCS. For example, system engineers may wish only to disclose the top level 'Claim' externally, therefore, the 'ArgumentPackageInterface' "OB.API1" is used which contains an 'asCited Claim' "G2" which will be referenced externally.

Similar case is applied to "TS.API1" 'ArgumentPackage', where the top level 'Claim' "G3" is cited in the 'ArgumentPackageInterface' "TS.API1". It is to be noted that both "OB.API1" and "TS.API1" contains their own argumentation elements including other sub-'Claims' and 'ArtifactReferences' to support the main 'Claim', which are not shown due to the complexity of the model structure.

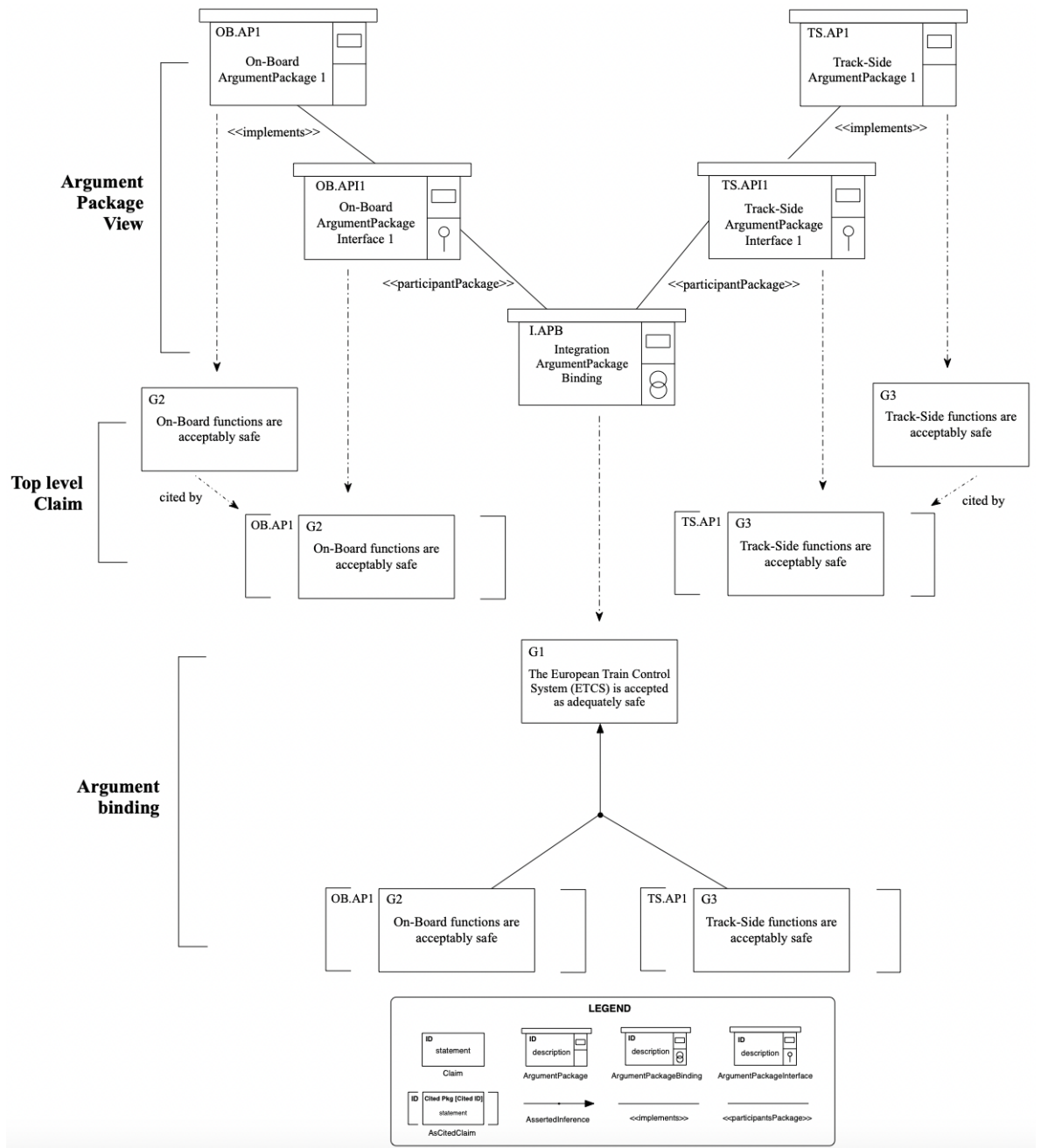


FIGURE 4.60: Example of 'ArgumentPackages'

To integrate "OB.API1" and "TS.API1", an 'ArgumentPackageBinding' named "I.APB" is created. "I.APB" contains an argument that specifically bind "OB.API1" and "TS.API1". Within the "I.APB", 'Claim' "G1" is asserted to argue the safety of ETCS. It is supported by 'Claims' "G2" and "G3". Note that "G2" and "G3" are 'asCited Claim' which cites 'Claim' "G2" in "OB.API1" and 'Claim' "G3" in "TS.API1".

Other than the 'ArgumentPackages', in SACM, there is an 'ArgumentGroup' that can be used to associate a number of argumentation elements to a common group, e.g., representing a common type of elements or purpose or being of interest to a particular user.

The ‘ArgumentGroup’ is visually represented using dotted-line where all the elements that are categorised as one group are placed within the dotted-line.

To show how the SACM ‘ArgumentGroup’ is used in an assurance case, we adopted an assurance case example from [91] that is about the multiple-view in safety cases. In [91], the safety case is presented using GSN which used a method called as a multi-view in order to show different views within a safety case. The example in [91] is reconstructed and presented using SACM notation as can be seen in Figure 4.61.

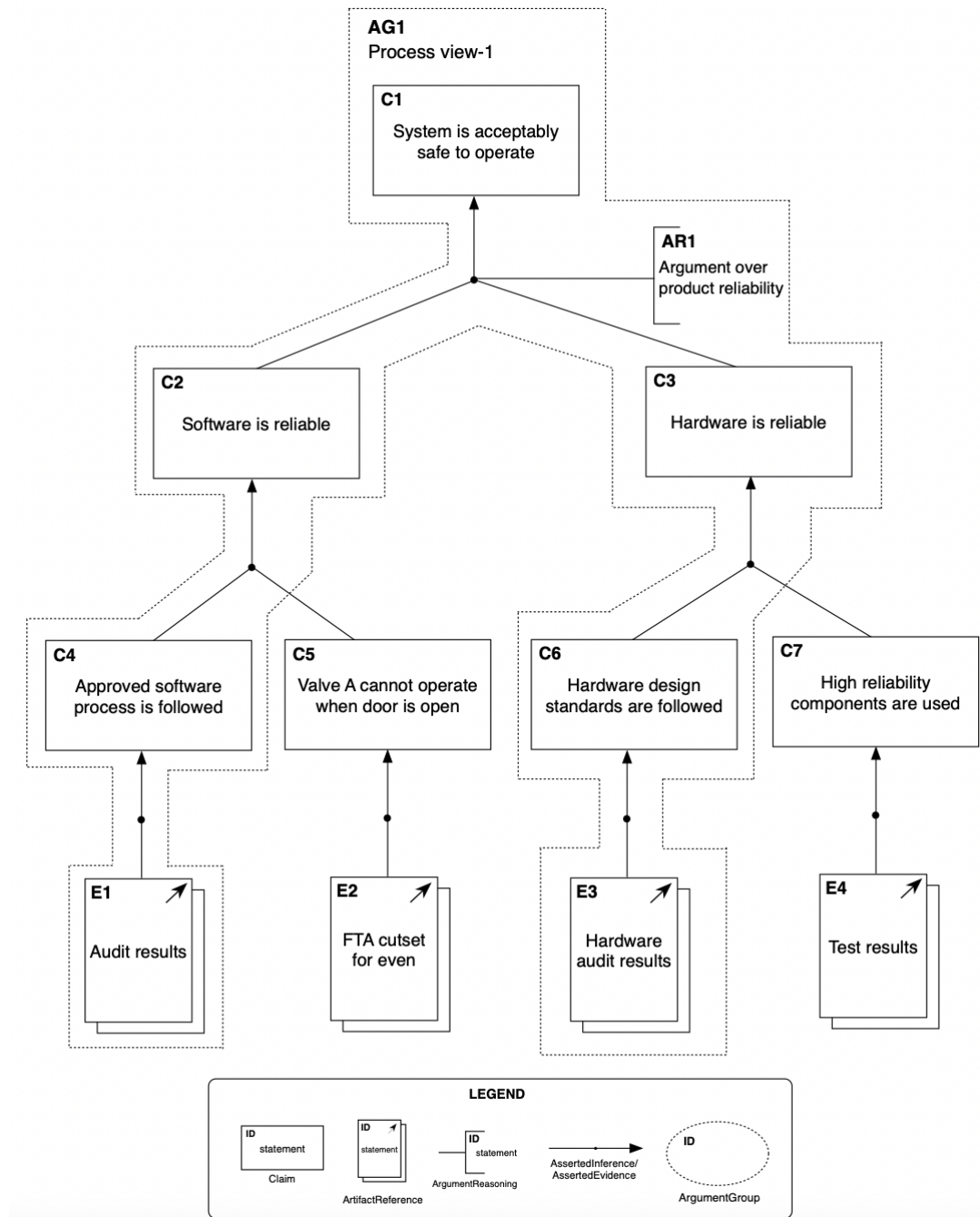


FIGURE 4.61: Example of ‘ArgumentGroup’

In Figure 4.61, 'ArgumentGroup' named "AG1" is presented to show a number of elements that are considered related each other. In this example, 'Claim' "C1", "C2", "C3", "C4", and "C6", together with 'ArgumentReasoning' "AR1", and 'ArtifactReference' "E1" and "E3", are categorised as one group to show elements that are related each others, specifically, about a process view presented in an assurance case.

In the next chapter, the evaluation regarding the proposed visual notation design approach is discussed.

Chapter 5

Evaluation of the design approach

5.1 Introduction

One of this thesis's contributions is to develop an approach for designing a visual notation based on a defined metamodel. In Chapter 3, the proposed approach was discussed. The application of the approach was also explained in Chapter 4 in a SACM argumentation notation development case study. In Chapter 5 and 6, the evaluation of this thesis is discussed.

The evaluation of this thesis consists of the following parts:

- The first part of the evaluation is related to evaluating the proposed visual notation design approach. The evaluation of the proposed approach is conducted through an empirical study involving novice users by comparing the effectiveness of the SACM notation as the notation developed using the proposed approach with an existing notation (GSN) that is developed with another approach. This evaluation is discussed in this Chapter (5).

Ideally, the proposed approach's evaluation is better to be done by asking another notation designer to use the proposed approach to create a notation. However, due to the difficulty in finding the participant as the notation designer, we decided to evaluate the proposed approach by comparing the resulting notation's effectiveness (developed using the proposed approach) with an existing notation that is developed using another approach.

- The second part of the evaluation is related to evaluating the effect of the design decision made during the SACM notation development. This evaluation is conducted through an empirical study involving another group of novice users by comparing the SACM notation's effectiveness with SACM alternate notation effectiveness developed using the proposed approach, however, with different design decisions with the SACM notation. By evaluating the design decision's effect as part of the proposed approach, we might understand how effective the design decision

made for the SACM notation. This evaluation is discussed in Chapter 6.

- The third part of the evaluation is related to seeking feedback from the experienced assurance case users such as, safety case developers, engineers, specialist, and managers related to the created SACM notation. We considered the experienced users' feedback towards the created SACM notation as a valuable input for further development of SACM notation. This evaluation is discussed in Chapter 6.
- The last part of the evaluation is related to applying the SACM notation in tool support. This evaluation shows that the created SACM notation can be implemented in tool support, for example, to develop assurance case models. This evaluation is discussed in Chapter 6.

Other than the above evaluations, the created SACM notation has also been evaluated by OMG committee. Based on their evaluation, the produced SACM notation has been included as part of the published version of the SACM 2.1. specification¹.

5.2 Study definition

5.2.1 Objectives

In this chapter, the evaluation of the proposed approach is described. The objective of the evaluation that is discussed in this chapter is to analyse the effect of the application of the proposed visual notation design approach when being used to develop an assurance case notation. In this case, the proposed approach was used to develop the SACM argumentation notation (also referred to as SACM notation). The produced notation is evaluated to determine if it is an effective notation.

In order to understand the effect of the application of the proposed approach, we conducted an experimental study by comparing the SACM notation, as the notation that is developed based on the proposed approach, and an existing assurance case notation that was not developed using this approach. In this case, GSN is selected as a comparative notation because of the following considerations:

- GSN was not developed using the proposed approach
- GSN is a notation that is in the same domain with SACM (assurance/safety case)
- GSN is one of the most widely adopted assurance case notation in industries and research.
- Much literature on GSN, including the GSN community standard, is publicly accessible and easily available.

¹<https://www.omg.org/spec/SACM/2.1/PDF>

In this thesis, the effectiveness of the resulting assurance case notation is used as a parameter to assess the effect of the application of the proposed visual notation design approach. The characteristics of an effective notation in this study can be described as follows (as previously discussed in section 3.1):

- **Accurate:** a notation that can be used by its user to construct a model (diagram) that is correct in every detail.
- **Easy to learn:** the visual representation of the elements should be easy to identify by its users so that the users can easily read the diagram that consists of multiple elements of the notation.
- **Easy to use:** the users can use (draw) the visual representation of the elements quickly when they need to create a diagram using the notation (this characteristic is derived from the definition of notation efficiency (from [3]) to measure the user's speed when using the notation to create a diagram).

The following characteristics of the notation were also observed as additional characteristics that are based on the participants' perception after learning and using the notation in an assurance case model construction task. Several survey questions were asked to the participants to investigate the following additional characteristics:

- **Ease of learning** - which notation that they think is easiest to learn after they reading the manual and receiving a tutorial regarding the notations.
- **Ease of use** - which notation that they think is easiest to use after they use the notation to create an assurance case model.
- **Intention to use** - which notation that they intend to use in the future when facing a similar problem (i.e. create an assurance case).

To summarise, we state the objective statement in the form prescribed by Wohlin et al. in [92] as the following:

Analyse the SACM notation and Goal Structuring Notation (GSN).

For the purpose of evaluating the proposed visual notation design approach that is used in the development of the SACM notation.

With respect to the effectiveness of the notation.

From the point of view of the University students.

In the context of assurance case model experiment (construction task and notation intuitiveness studies).

5.2.2 Experimental objects

The experimental objects of this study were:

- The SACM argumentation notation that was developed by using the proposed visual notation design approach.
- The Goal Structuring Notation (GSN) as a comparative notation that was developed using another notation design approach.

5.2.3 Perspective

The experiment was conducted from the perspective of University students majoring in Computer Science and Information Technology with no prior knowledge and experience related to the assurance case modelling and notations. These participants were chosen since they could provide an objective opinion and reduce bias as they have no prior knowledge and experience related to the assurance case modelling and notations.

5.3 Planning

Based on the objective of the study, we considered the following research questions:

1. Which notation is more effective in terms of its accurate used by the participant to construct an assurance case model?.
2. Which notation requires less effort in terms of time required by the participant to create an assurance case model?
3. How effective is the produced SACM notation in the sense of the intuitiveness of the visual appearances of the element?
4. How effective is the produced SACM notation in the sense of its intuitiveness related to the identification of the correct variants of the element by the participant?
5. Which notation that the participants' considered is relatively easy to learn?
6. Which notation that the participants' considered is relatively easy to use?
7. Which notation that the participant intends to use in the future when facing similar problems (create an assurance case)?

5.3.1 Participant selection criteria

The participants were selected through a non-probabilistic sampling approach which means the probability of selecting each experimental subject is unknown. The non-probabilistic sampling technique used in this experiment is convenience sampling, i.e., the most convenient persons are selected as subjects. In this study, the invited participants were the university students majoring in Computer Science and Information Technology from Telkom University and Yarsi University. Both Universities are located in Indonesia.

5.3.2 Context selection

The university students in this experiment were referred to as "participants". This experiment was supported by Telkom University and Yarsi University. Both Universities helped to gather the students who are willing to participate voluntarily in this experiment. The information regarding this experiment was announced by the Department of Computer Science and Information Technology in both universities. A classroom equipped with PCs for the participants to be used in this experiment was also provided by the universities.

Before the experiment began, all the participants were asked to answer questions regarding their background and experience such as their domain of work (e.g. Student), and experience related to assurance case modelling and notations. We considered the second question necessary to clarify that they have no prior knowledge and experience related to the assurance case modelling and notations.

5.3.3 Hypothesis formulation

The hypothesis was formulated to compare the effectiveness of the SACM notation and the GSN in the assurance case model construction task. This was done by comparing the accurate use of the notation by the participants to create an assurance case model. We also compared the average time spent by the participants to create an assurance case model using SACM notation and GSN to evaluate the easy to use characteristic of the notation.

We formulated the hypothesis in this study by following the thesis formulation guideline based on the book of *Experimentation in Software Engineering* [92]. The following is the formulated hypothesis for the experimental study related to the effectiveness of the notation in the context of the accurate use of the notation by its users to create an Assurance Case model ('AC'):

- $H_{0AC}: AC_{GSN} \geq AC_{SACM}$
- $H_{1AC}: AC_{GSN} < AC_{SACM}$

The ' AC_{SACM} ' represents the score (average correct answer) in using the correct visual representation to represent the SACM element in an assurance case model construction task. The ' AC_{GSN} ' represents

the score in using the correct visual representation to represent the GSN element in an assurance case model construction task. 'H0' is the null hypothesis, which is true when the average score from total participants using the SACM notation to create an assurance case model is equal or less than using the GSN. 'H1' is the alternative hypothesis (expected), which is true when the average score from total participants using the GSN to create an assurance case model is less than using the SACM notation.

The following is the formulated hypothesis for the experiment used to compare the easy to use characteristics of the SACM notation and GSN, in terms of Time Spent ('TS') to create an assurance case model using a particular notation:

- $H0_{TS}: TS_{GSN} \leq TS_{SACM}$
- $H1_{TS}: TS_{GSN} > TS_{SACM}$

' TS_{GSN} ' represents the average time spent by the participant to create an assurance case model using GSN. ' TS_{SACM} ' represents the average time spent by the participant to create an assurance case model from the same case study using SACM notation. 'H0', null hypothesis, is true when the average time required to create an assurance case model using SACM notation is greater or equal to the average time required to create an assurance case model using GSN. 'H1', the alternative hypothesis (expected), is true when the time spent to create an assurance case model using SACM notation is less than the time spent by the participant to create an assurance case model using GSN.

5.3.4 Variable selections

Dependent variables: the effectiveness of the notation in the sense of the accurate use and easy to use characteristics of the notation by its users.

Independent variables: the SACM notation and GSN.

Environmental variables: the learning speed of the participants.

5.3.5 Study design

The participants were divided into two groups by considering their time availability to participate in the study. Each group received different treatments in terms of the sequence of the tasks that the participants involved during the study. Table 5.1 shows the planned design for the experimental study.

First of all, the participants were asked to read the study objective and the consent form. In the consent form, their participation was explicitly stated as voluntary in this study; therefore, they may refuse to take part in this study or exit at any time without penalty. The participants were also allowed to decline to answer any particular question that they do not wish to answer for any reason. However,

TABLE 5.1: Study design

Group A	Group B
Introduction and study information	
Ethics and data privacy explanation (consent form)	
Demographics	
Part 1: Visual representation intuitiveness	
Part 2: 2.1. Read and received basic SACM notation tutorial 2.2. Create an assurance case model using SACM notation 2.3. Identify SACM element variants 2.4. Read and received basic GSN tutorial 2.5. Create an assurance case model using GSN 2.6. Identify GSN element variants	Part 2: 2.1. Read and received basic GSN notation tutorial 2.2. Create an assurance case model using GSN notation 2.3. Identify GSN element variants 2.4. Read and received basic SACM tutorial 2.5. Create an assurance case model using SACM 2.6. Identify SACM element variants
Part 3: Visual variables utilisation in a notation	
Part 4: Participant's perception towards the notations	
Participants' feedback	

we still count their responses to the questions that they chose to answer, and we included the gathered responses in the data analysis. In this consent form, the data being gathered in this study and where the data being stored were also being explicitly stated.

After receiving information regarding the objective of the study, all the participants were expected to answer several questions related to the demographic information. The gathered demographic data was used to ensure that the participants who involved in this study are relevant.

In the first part of the study, two survey questions were asked to the participants regarding the intuitiveness of the SACM visual representations. The intuitiveness of the notation is also known as the semantic transparency of a notation (i.e. the use of visual representations of a notation whose appearance suggests their meaning). According to this definition, there are two elements of the SACM notation that were designed as an icon-type visual representation: 'ArtifactReference', and 'ArgumentPackage'. An icon-type visual representation can be used to represent the semantic meaning of the visual representation [3]. In this part, we asked the participants to assess the intuitiveness of the 'ArtifactReference', and 'ArgumentPackage'. The participants were given a figure (visual representation) of a particular SACM notation, and then they were asked to identify the most suitable description regarding the given visual representation. They only can choose one answer to the given answer options. They also can type their answer in the case they could not find the answer from the given options. In this part, we were not evaluating the notation intuitiveness of the GSN because there is no GSN elements were found to be designed as an icon-type visual representation.

The second part of the study consists of several sessions with different objective and activity.

In session 2.1, the participants in Group A were handed a printed copy of the basic SACM notation documentation. They were expected to read this documentation in approximately ten minutes. After that, they received a tutorial related to the SACM notation. Participants in Group B were handed a printed copy of the basic GSN documentation. They were also expected to read it in approximately

ten minutes, and after that, they received a tutorial regarding the GSN notation. After the participants in both groups received the tutorial, the experiment facilitators were collecting the printed copy of the SACM and the GSN documentation.

In session 2.2., the participants were handed a printed copy of the case study document, pen and pencil, and A4 papers. They were informed to participate in an activity of an assurance case model construction based on the notation that they have just learned. The participants could not read the notation documentation in this session because it was handed back to the experiment facilitators before this session begin. The participants were told that the timing of their activity in constructing the assurance case model is recorded. Therefore, for them who finished creating the assurance case model, they need to immediately hand the created model to the facilitators in order to record the timing. The participants in Group A were asked to create an assurance case model based on the given case study using the SACM notation, and the participants in Group B using the GSN.

In session 2.3., the participants in Group A were asked to answer several survey questions related to the identification of the SACM element variants by observing the visual representation similarities between the given elements. Similarly, the participants in Group B were asked to answer several questions related to the identification of the GSN element variants.

In session 2.4., the participants in Group A were handed a printed copy of the GSN documentation, and the participants in Group B were handed a printed copy of the SACM notation documentation. They were asked to read the documentation in approximately ten minutes, and after that, they received a tutorial regarding the notation. The documentation was handed back to the facilitators after the tutorial finished in this session.

In session 2.5., the participants in both groups were asked to create another assurance case model based on the same case study that they used previously in the session 2.2. However, in this session, the participants in Group A were asked to create the assurance case model using the GSN, and the participants in Group B were asked to use the SACM notation. They were also told that they need to hand the created assurance model immediately to the facilitator after they finished it in order to record the timing.

In session 2.6., the participants were asked to answer several survey questions related to the identification of the element variants. For participants in Group A, they were asked to identify the variants of the GSN elements, and the participants in Group B were asked to identify the variants of the SACM elements.

In part 3, all the participants in both groups were asked to answer several survey questions related to the utilisation of the visual variable in visual notation design. This is an extra part of the experiment that is designed to understand more about the intuitiveness of the visual representation that is created based on the combination of the visual variables, especially, in the context of arrow direction, line

size (thickness), and brightness.

In part 4, all the participants in both groups were asked to answer questions related to their perception of the notations that they just have learned and used during the experiment. After that, the participants were expected to provide their feedback related to the assurance case notations and the experiment activity in general.

The experiment was conducted in a classroom at Telkom University and Yarsi University - Indonesia, that is equipped with PCs for the participants to use to access the online experiment platform. The web address to the online platform was given to the participants just before the experiment begins. All the experiment information and the experiment questions were available on the online platform.

In order to identify the potential problems related to the experiment and to make sure that the experiment will run as planned, a pilot study was conducted prior to the experiment. The result from the pilot study was used to refine to design the experiment.

5.3.6 Instrumentation

The following materials were provided to the participants during the study:

- A printed copy of the SACM notation documentation. This document describes the basic notation of the SACM and how to use them to develop an assurance case model. The SACM elements that we considered as the basic elements including 'Claim', 'ArtifactReference', 'ArgumentReasoning', 'AssertedInference' relationship, 'AssertedEvidence' relationship, and 'AssertedContext' relationship. Other SACM elements were excluded due to consider the duration of the experiment that was relatively limited, and the learning load of the participants in a relatively short time.
- A printed copy of the GSN documentation. This document describes the basic notation of the GSN and how to use them to develop an assurance case model. The elements that were included in this documentation, including the GSN Goal, Strategy, Context, Solution, SupportedBy relationship, and InContextOf relationship. Other elements such as Assumption and Justification were excluded in this documentation since these elements were considered the variants of an element in SACM notation.
- A printed copy of case study instruction. This document provides a fragment of an assurance case that was adopted from [25] used as a case study in the assurance case construction task that presented using a tabular approach in this study.
- Questions related to the experiment that was provided in an online survey platform. The order of the questions was designed differently for two different groups in this experiment. Therefore, each group was provided by two different address to access the online survey platform.

5.4 Operation

5.4.1 Preparation

All the experiment materials were prepared to be used for the experimental study, including ethics and data privacy forms (consent forms). An ethics and data privacy application form have been submitted and approved by the University Ethics Committee before the experimental study since it involved human subjects.

A pilot study was conducted before the experiment to stimulate the experiment environment. Four students were invited to participate in a pilot study. They were divided into two groups considering their time availability to participate in the pilot study. The result from the pilot study was used to refine the experiment materials. For example, some statements were added to the case study instruction document as a reminder to the participants to write down all the assurance case statements (text) to the diagram that they develop during the assurance case model construction task.

5.4.2 Execution

The participants were involved in the experiment as planned. First, they read the experiment information and the consent form. After that, they answered the demographic questions and the notation design intuitiveness questions. Then, the participants read and received the tutorial regarding the notations, and they also completed the assurance case model construction task. They also answered the questions related to the variants of the visual representation and the visual variable utilisation in notation design. The participants also provide feedback regarding the notation and the experiment activity.

5.4.3 Data validation

All the participants from the pilot study were identified never have prior knowledge and experience related to the assurance case modelling and notations. This information was confirmed after they answered the demographic questions during the pilot study. Their feedback from the pilot study is then relevant to be used as an input to refine the experiment design. The participants from the pilot study were not allowed to be involved in the main experiment to reduce bias.

From the demographic data gathered in the experiment, all the participants are confirmed never have prior knowledge and experience related to the assurance case modelling and notations. Therefore, we considered the data gathered from the participants in this experiment are valid and relevant to the study objectives.

5.4.4 Data collection

The experiment data were collected through an online survey platform used by the participants to answer the questions related to the experiment.

For Group A participants: <https://selviandro.limequery.com/587758?lang=en>

For Group B participants: <https://selviandro.limequery.com/555827?lang=en>

The data related to the assurance case model construction were collected manually during the experiment since the participants creating the model manually using pen and papers.

The data were collected from participants from Telkom University and Yarsi University. Both universities are located in Indonesia. In total, there were 82 participants involved in this experiment, 13 of them were from Telkom University, and 69 from Yarsi University. For students from Telkom University, the experiment was conducted on the 28 November 2019 in the morning for Group A, and in the afternoon for Group B. There were 6 participants in Group A and 7 participants in Group B. For students from Yarsi University, the experiment was conducted on 9 December 2019 for Group A with a total of 43 participants. For Group B, the experiment was conducted on 12 December 2019 with a total of 26 participants.

5.5 Data analysis and interpretation

5.5.1 Demographics

We first established some demographic data to ensure that the collected data were relevant to this study. We asked several questions to the participants regarding their background and experience.

The first question was about the participant's domain of work. This is a multiple-choice question with several answer options provided. The participants were allowed to choose more than one answer that applied to their condition, and they were also allowed to submit an additional answer if their domain of work was not provided in the answer options.

From total 82 participants involved in this experiment, 77 participants (46 from Group A, and 31 from Group B) answered "Student" as their domain of work, 4 participants (2 from each group) answered "Academia", 2 participants answered "Software Engineer" (1 participant from each group), and 1 participant (from Group B) answered "IT Professional" as their domain of work. There was no participant answered 'Other' option or not completing this question.

During the experiment, we confirmed to the participants that they were all university student, and only a few of them work as a part-timer, for example, as a teaching/research assistant (academia), IT support, and software engineer at the university. Hence, the total responses received related to this

question (84 responses) is more than the total participants involved in this experiment (82 participants) since there were participants who have more than one work domains.

As for the second question, we asked the participants about their experience related to the software/system modelling and notations. From total 82 participants, the majority of the participants, 93% or 76 participants (44 from Group A, 32 from Group B), claimed that they have experience in system/software modelling and notations. Only 7% or 6 participants (5 from Group A, 1 from Group B) who have no prior experience related to software/system modelling and notations.

As a follow-up question for the participant who has experience in software modelling and notations, we asked which notation they have experience with. Most of the participants, 76 participants (44 from Group A, 32 from Group B), have experience with Entity-Relationship Diagram (ERD), 20 participants (all from Group A) claimed that they have experienced with Flowchart, 11 participants (6 from Group A, and 5 from Group B) claimed that they have experienced with UML notation.

From the gathered data related to the participants experience on the modelling language and notation, we learned that not all the participants have experience in software modelling and notation even though the experiment was announced to be available for computer science/information technology/information system students. We assumed it was possible that few of the participants still a first-year student who has not yet learn about modelling languages. However, this result was not affecting this experiment's result since experience in modelling language and notation was not the requirement for the participant to be involved in this experiment.

In the next question, we asked about the participant's knowledge and experience related to the assurance case modelling and notations. There were no participants claimed that they have prior experience related to the assurance case modelling and notations. We received 82 responses (100%) that stated they have no experience related to assurance case modelling and notations.

To summarise, based on the gathered demographics data, we considered all the participants relevant to this study because they have no prior knowledge and experience in assurance case modelling and notations. This was our mandatory requirement for the participants to involve in this experiment.

5.5.2 Visual representation intuitiveness

In this part, the result of the study related to the following question is described:

How effective is the produced SACM notation in the sense of the intuitiveness of the visual appearances of the element?

Two questions were asked to the participants to understand how effective the produced SACM notation is in the context of the intuitiveness of the element's visual appearances.

In the first question, the participants were asked to identify the most suitable description for the visual representation of the ‘ArtifactReference’ element. In the second question, the participants were asked to identify the most suitable description for the ‘ArgumentPackage’ element’s visual representation. These two elements (‘ArtifactReference’ and ‘ArgumentPackage’) are designed based on icon-type visual representation. The elements designed based on icon-type visual representation should represent the meaning of the element in its visual appearances. For both questions, the participants were given the information regarding the element’s visual representation (‘ArtifactReference’ for the first question, and ‘ArgumentPackage’ for the second question). Several answer options were provided for the participants to choose. They can also provide their answer if they think there is no suitable answer from the given options. If the participant chose the correct answer of the meaning of a particular SACM element, then the element’s visual representation is considered intuitive.

For the first question, we received a total of 81 responses from both group A and B. As illustrated in Figure 5.1, the highest response received from the participants (49% or 40 participants) was the correct answer to describe the meaning of the ‘ArtifactReference’, in this case, they chose: *"A document/artifact icon that can be used as a reference to evidential or contextual information to support a structured argument"*, as an answer for the most suitable description regarding the ‘ArtifactReference’ visual representation:

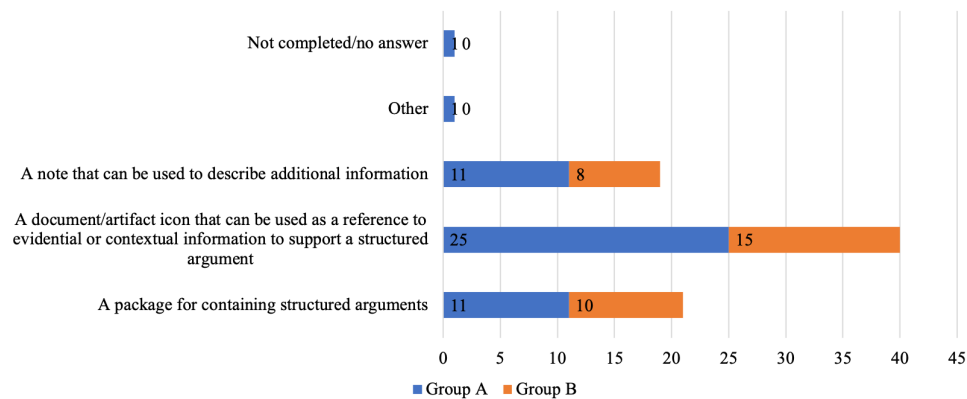


FIGURE 5.1: The participant’s responses regarding the ‘ArtifactReference’ visual representation intuitiveness question

The remainder of the participants chose *"a package for containing structured arguments"* (26% or 21 participants), and *"a note that can be used to describe additional information"* (24% or 19 participants) as their answer. Only one participant (1%) who chose *"Other"* option, and provide irrelevant input to the given question. Other than 81 responses received related to this question, there was also a participant that chose not to answer this question.

Based on the first question’s responses, we considered an improvement is necessary to enhance the intuitiveness of the ‘ArtifactReference’ visual representation. Consider less than 50% participants were able to identify the correct description for the ‘ArtifactReference’ visual representation.

As for the second question, we received a total of 80 responses from both groups. As can be seen in Figure 5.2, the highest response received (46% or 37 participants) was: *"A note that can be used to describe additional information"*. However, this was not the correct answer to describe the meaning of the ‘ArgumentPackage’. The second-highest responses, 32% or 25 participants, was: *"A package for containing structured arguments"* - this was the correct answer to describe the meaning of ‘ArgumentPackage’.

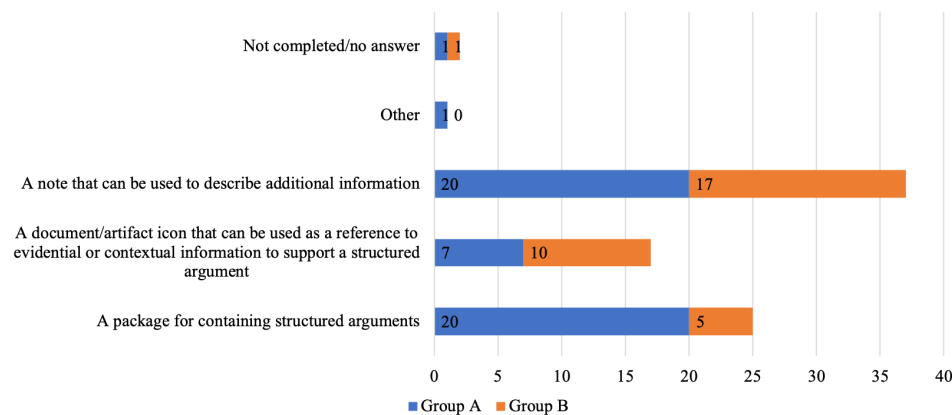


FIGURE 5.2: The participant’s responses regarding the ArgumentPackage visual representation intuitiveness

The remainder of the participants chose *"A document/artifact icon that can be used as a reference to evidential or contextual information to support a structured argument"* as their answer (21% or 17 participants), and there was one participant (1%) who chose *"Other"* option. However, we considered the text input response regarding the *"Other"* option was not relevant to this question. Other than 80 responses that we received, there were also 2 participants who chose not to answer this question.

Based on the participants’ responses, the ‘ArgumentPackage’ visual representation can be considered less intuitive than the result we received in the first question. In the first question, the highest response received was the correct description of the element. In the second question, the highest response received was not the correct description of the element. Although the second-highest responses reflect that the participants chose the right answer to describe the meaning of the ‘ArgumentPackage’ visual representation, we considered further design improvement is required for the ‘ArgumentPackage’ visual representation.

5.5.3 Accurate use of the notation

In this part, we described the result of the study related to the following question:

Which notation that is more effective in terms of its accurate use by the participants to construct an assurance case model?

To answer the above question, we designed and conducted an assurance case construction task for all Group A and B participants.

Group A participants learned about the SACM notation first and used it to create an assurance case model based on the given case study. After that, they learned about the GSN and used it to create the same case study.

Group B participants learned and used the GSN first to create an assurance case model, and then learned used the SACM notation to create another model using the same case study.

The model construction task was designed to provide different treatments to each group to identify the effect of the order of the notation that they learnt concerning the effectiveness of the notation in the context of its accurate use to create a model.

The following materials were provided for the participants to use in this study:

- SACM notation (basic elements) documentation.
- GSN (basic elements) documentation.
- Case study instruction.

The participants were handed a printed case study instruction that provides an assurance case about the safety aspect of a control system adopted from the GSN community standard. The adopted assurance case was presented in a tabular presentation, as shown in Figure 5.3. The participants were asked to create the diagram representation of the case study using the notation they just learnt. In the case study instruction, the participants were also told to use a Diamond shape to indicate that they could not remember an element's particular visual representation.

Main Claim	Contextual information	Supporting Claim	Contextual information	Reasoning	Supporting Claim	Reference to Evidence
[G1] Control System is acceptably safe to operate	[C1] Operating Role and Context	[G3] Software in the Control System has been developed to SIL appropriate to hazards involved	[C4] Hazards identified from FHA (Ref Y)	[S2] Argument over allocated SIL for Primary and Secondary elements	[G7] Primary Protection System Developed to SIL 4	[Sn3] Process Evidence for SIL4
	[C2] Control System Definition		[C5] SIL Guidelines and Process		[G8] Secondary Protection System Development to SIL 2	[Sn4] Process Evidence for SIL2

FIGURE 5.3: Tabular assurance case used as a case study in the assurance case model construction task

We constructed the correct representation of the assurance case based on the case study using both GSN and SACM notation, which can be seen in Appendix B as a benchmark to identify the element's correctness.

If the case study is constructed using SACM notation, there are in total of 20 SACM elements with details as follow:

- 4 ‘Claim’ elements (Claim G1, G3, G7, and G8)
- 1 ‘ArgumentReasoning’ (S2)
- 6 ‘ArtifactReference’ elements (4 as Context (C1, C2, C4, and C5), and 2 as Evidence (Sn3 and Sn4))
- 3 ‘AssertedInference’ elements (relationship between G3 and G1, G7 and G3, G8 and G3)
- 2 ‘AssertedEvidence’ elements (relationship between Sn3 and G7, Sn4 and G8)
- 4 ‘AssertedContext’ elements (relationship between C1 and G1, C2 and G1, C4 and G3, C5 and G3)

If the case study is constructed using GSN, there are in total of 21 GSN elements with details as follow:

- 4 Goal elements (Goal G1, G3, G7, and G8)
- 1 Strategy (S2)
- 2 Solution elements (Sn3 and Sn4)
- 4 Context elements (C1, C2, C4, and C5)
- 6 SupportedBy elements (relationship between G1 and G3, G3 and S2, S2 and G7, S2 and G8, G7 and Sn3, G8 and Sn4)
- 4 InContextOf elements (relationship between G1 and C1, G2 and C2, G3 and C4, G3 and C5)

For each created assurance case model, we calculated its score by identifying the correctness of each element. For each element, we give 1 point if the element’s visual representation is drawn correctly and another 1 point if the element’s positional placement is correct in the diagram. We then normalised the maximum score for each created model to 100. In the case of a model created using the SACM notation, the participant will get a 100 score if they correctly created the visual representation and the positioning of all 20 elements in the model. Similarly, for the model created using the GSN, the participant will get 100 if they correctly created the visual representation and the positioning of all 21 elements in the model.

From the result of the assurance case model task, we compared the calculated average score of the assurance case models created using the SACM notation by all the participants from Group A and

B, with the average score of the assurance case models that were created using the GSN by all the participants from both Groups.

Table 5.2 shows the calculated average score from both Groups in creating an assurance case model using SACM notation and GSN.

TABLE 5.2: Average score of the assurance case construction task using SACM notation and GSN

		Average Score	
		SACM notation	GSN
Group A		58.71(36 participants)	75.4(36 participants)
Group B		58.13(32 participants)	59.88(33 participants)
Total average(Group A + B)		58.71(68 participants)	67.98(69 participants)

As can be seen in Table 5.2, the total average score of the participants from both Group A and B in creating the assurance case model using GSN (67.98) is greater than the total average score of the participants from both Group A and B in creating the assurance case model using SACM notation (58.71). This result indicates that the GSN is more effective in terms of its accurate used by the participants from both Group A and B when being used to create an assurance case model, especially, relative to the accurate use of the SACM notation by the participants to construct an assurance case model from both Groups.

According to the data presented in Table 5.2, both Group A and B participants received a better score in creating assurance case using GSN than using SACM notation. This result indicates that learning order might not affect the performance of all the Group A and B participants.

Table 5.3 summarises the assurance case model construction task result based on Group A and B participants' performance using SACM notation and GSN.

TABLE 5.3: Assurance case model construction task result (SACM notation Vs GSN): group comparison

	Group A		Group B	
	SACM	GSN	GSN	SACM
Model received	36	36	33	32
Average score	59.24	75.4	59.88	58.13
Max score	100 (3 participants)	100 (15 participants)	100 (7 participants)	100 (3 participants)
Min score	5 (1 participant)	21.4 (1 participant)	9.52 (1 participant)	15 (1 participant)
Most accurately drawn element	Claim (58%)	Goal (67%) Context (67%)	Strategy (55%)	Claim (47%)
Most inaccurately drawn element	ArtifactReference (as Context 11% as Evidence 19%)	SupportedBy (44%)	SupportedBy (24%)	ArtifactReference (as Context 19% as Evidence 22%)

As presented in Table 5.3, beside the participants' average score from both groups using GSN is greater than using SACM notation, we also identified six participants in total (3 from Group A and 3 from Group B) who get a perfect score (100) in creating assurance case model using SACM notation. As for the GSN, we identified 22 participants who get a perfect score (100) in creating their model.

This result might indicate that the accurate use of the GSN by the participants in the assurance case model construction task is relatively better than the SACM notation.

Based on the participants' performance in the assurance case model construction task, SACM notation 'Claim' was found as the type of element that most accurately drawn by Group A and B participants. Among other 'Claim' elements that must be drawn correctly by participants in the model, 'Claim' G1 (main 'Claim' in the model) was found to be the most accurately drawn by the Group A participants, and 'Claim' G3 for Group B participants. Common mistakes found regarding the drawing of 'Claim' elements found, for example, drawn as an 'ArtifactReference' and drawn as diamond shape (to indicates the participant could not remember the visual representation of 'Claim').

The 'ArtifactReference' was found as the type of SACM notation element that both Group A and B participants struggled to draw correctly, especially the 'ArtifactReference' used as a Context. The majority of error found related to the drawing of 'ArtifactReference' was related to the use of arrow as part of the 'ArtifactReference' visual representation. Both Group A and B participants struggled to remember that the arrow must also be drawn as part of the 'ArtifactReference' visual representation. The arrow is designed to represent the meaning of a reference in the context of 'ArtifactReference' visual representation. Other common errors found, for example, drawn as a rectangle, diamond, and no drawing.

As for the GSN, the Goal ('Claim' in SACM) and the Context elements were found as the type of GSN element that most Group A participants accurately draw, and the Strategy element for Group B participants. Among other Goal elements that must be drawn correctly by the participants in the model, Goal G1 was found as the most accurately drawn by the Group A participants, and Goal G7 and G8 as the elements that Group A participants most struggled to draw. As for the GSN Context, Context C1 and C2 were found more accurately drawn by the participants than Context C4 and C5. Some participants did not draw anything to represents Context C4 and C5.

The GSN SupportedBy relationship was found as the type of element that most Group A and B participants struggled to draw accurately. Wrong direction and the wrong type of arrowhead were found as the common error committed by the participants in drawing the GSN SupportedBy relationship. Some participants were found drawing the GSN SupportedBy relationship using an arrowhead line without the solid-type of the arrowhead.

The result regarding the other elements of SACM notation and GSN can be read further in Appendix B.

Hypothesis testing

In this chapter, previously, we have formulated the hypothesis to evaluate the effectiveness of the

SACM notation and GSN in the context of the accurate use of the notation by the participants. Statistical hypothesis testing using a Two-Sample t-Test (for unequal variances) was conducted to evaluate the effectiveness of the SACM notation and GSN in the context of the participants' accurate use of the notation. Before conducting the t-Test, we identified the distribution of the data. The data distribution for Group A and B while using SACM notation and GSN is shown in a histogram as presented in Figure 5.4.

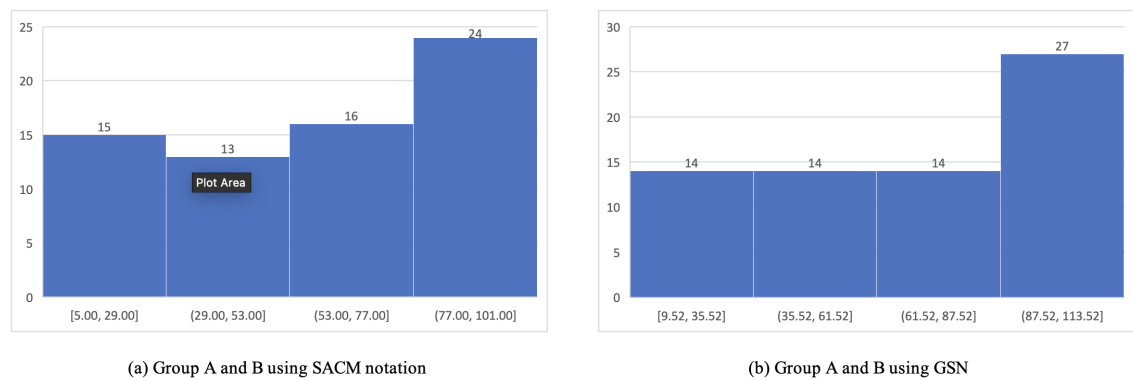


FIGURE 5.4: Data distribution Group A and B for the accurate use of the notation study (using (a) SACM notation and (b) GSN)

As presented in Figure 5.4, the data from Group A and B while using SACM notation and GSN was found non-normally distributed. However, based on the literature such as in [93], the distribution of the data will not be affecting the validity of the t-Test. Furthermore, the result of the t-test is presented in Table 5.4.

TABLE 5.4: t-test results for the accurate use of the notation study

Group	Notation	N	Mean	Variance	P value	Alpha
A+B	SACM	68	58.71	760.54	0.03	0.05
	GSN	69	67.98	953.30		

The first column of Table 5.4 indicates the groups participating in this study. The second column indicates the notations used by the participants to create the assurance case model. "N" in the third column indicates the number of participants involved in the study. In this case, 68 participants in total (from Group A and B) were involved in the assurance case model construction task using SACM notation, and 69 participants who used GSN. The "Mean" column indicates the resultant mean from participants' score in the assurance case model construction task. The "Variance" column indicates the variance from the gathered data of the participants' score in using each notation. Since the variance from the data using the SACM notation and GSN is unequal, the Two-Sample t-Test for unequal variances was used to statistically evaluate the data. The column "P-value" indicates the P-value associated with the mean generated from the participants' score using each notation. The "Alpha" column indicates the alpha number (margin of error), in this case, is 0.05.

As shown in Table 5.4, the assurance case model construction task's average score using SACM notation was 58.71 and 67.98 as the average score using the GSN. This means the accurate use of

the GSN is better than SACM notation (in this case, the result indicates something that we are not expected). Therefore, we need to test whether the Null Hypothesis can be statistically accepted (Null Hypothesis stated that accurate use of GSN is better than SACM notation). By conducting the t-Test, the resulting P-value is equal to 0.03, in this case, it is lower than the Alpha number that we used in this test(0.05), and it is not in the rejection region. Therefore, we may conclude that we can statistically accept that the GSN is more effective than SACM notation in the context of accurate use of the notation by the participants based on the assurance case construction task.

5.5.4 Notation that is easy to use

In this part, the result of the study related to the following question is described:

Which notation that requires less effort in terms of time required by the participants to create an assurance case model correctly?

The data used to answer the above question is the time data that the participants spent to create an assurance case model using a particular notation in the assurance case model construction task.

The details of Group A and B's performance results concerning the times spent by the participants in creating the assurance case models using both GSN and SACM notation are presented in Appendix B.

There were 68 models created by the participants from both Group A and B using SACM notation. As presented in Figure 5.5, the average time spent by the participants from both Group A and B to create the assurance case model using SACM notation was 10:42, with the longest time take was 15:44, and 05:16 for the shortest. As for the participants' models from both Group A and B using GSN, we received 69 models in total. From all 69 models created using GSN, the participants' average time was 10:48, with the longest, was 19:05, and 05:40 for the shortest.

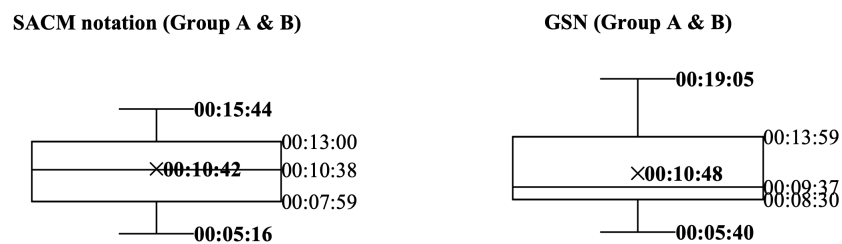


FIGURE 5.5: Performance results of Group A and B with respect to the times spent in creating the assurance case models using both notations (in hh:mm:ss)

Based on data presented in Figure 5.5, the result shows that the participants relatively quicker in creating assurance case model using SACM notation than using GSN, although the difference is only 0.06 seconds.

We also calculated the average time taken by the participants who get a perfect score in creating the assurance case models using SACM notation and GSN as summarised in Table 5.5.

TABLE 5.5: Timing data: average time spent by the participants from both Group A and B to create the correct assurance case models using SACM notation and GSN

	SACM notation	GSN
Number of models with a perfect score	6	22
Average time spent	00:09:04	00:09:19

Based on the data presented in Table 5.5, there are not much time differences between participants who get a perfect score in creating assurance case models using SACM notation than using GSN: the difference only 15 seconds. In this context, the participant's performance slightly quicker using SACM notation than GSN, however, the number of participants who create the assurance case model correctly using GSN is greater than the number of participants who create the assurance case model correctly using SACM notation.

We also identified that the participants tend to spend more time creating the assurance case model using the first notation they learned during the experiment. After they have experience with the first notation, they tend to spend less time in constructing the model using the second notation that they just have learnt. This is indicated by the average time spent by the participants in Group A in creating the assurance case model using the SACM notation (as the notation that they learned first during the experiment) is greater than the average time spent by them in creating the models using GSN (as the second notation that they learned during the experiment). As shown in Figure 5.6, the Group A participants relatively quicker when using GSN notation than using SACM notation with the average time using GSN 08:38 and the average time using SACM notation is 11:22.

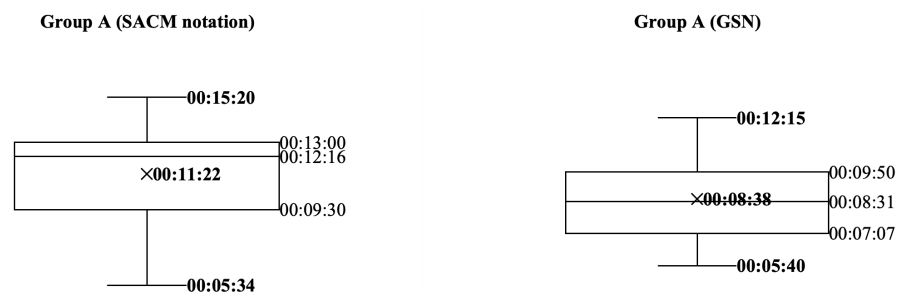


FIGURE 5.6: Performance results of Group A with respect to the times spent in creating the assurance case models using both notations (in hh:mm:ss)

A similar pattern was found in assurance case model construction by Group B participants, as can be seen in Figure 5.7, Group B participants relatively quicker in creating assurance case using SACM notation (as the second notation that they learned during the experiment) than using GSN (as the first notation that they learned during experiment). The participants' average time when using SACM notation is 09:57, and the average time spent by the participants when using GSN is 13:09.

Hypothesis testing

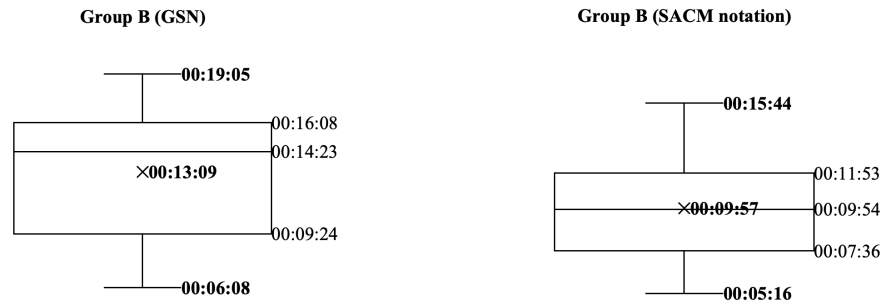


FIGURE 5.7: Performance results of Group B with respect to the times spent in creating the assurance case models using both notations (in hh:mm:ss)

In this chapter, previously, we have formulated the hypothesis to evaluate the effectiveness of the SACM notation and GSN in the sense of the time required by the participants to create an assurance case model.

The data distribution from the effectiveness of the notation study related to the time required by the participants to create an assurance case using the SACM notation and GSN is shown in Figure 5.8 using a histogram.

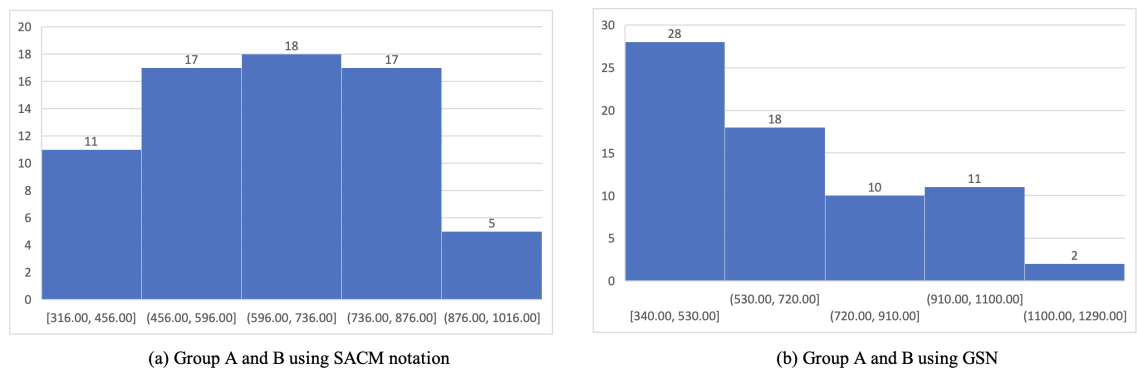


FIGURE 5.8: Data distribution Group A and B for the effectiveness of the notation study - time required to create an assurance case (using (a) SACM notation and (b) GSN)

Similar to the hypothesis testing to evaluate the accurate use of the notation, a Two-Sample t-Test for unequal variances was conducted to evaluate the effectiveness of the SACM notation and GSN in the sense of the time required by the participants to create an assurance case model. Although the distribution of the sampling data shows two different shapes of distributions: normal and skewed, based on the literature ([93]), it will not affect the validity of the t-Test. Furthermore, the result of the test is presented in Table 5.6.

TABLE 5.6: t-test results for the effectiveness of the notation in terms of time required by the participants to create an assurance case

Group	Notation	N	Mean (in seconds)	Variance	P value	Alpha
A+B	SACM	68	641.62	25425.31	0.43	0.05
	GSN	69	647.72	49594.82		

The first column of Table 5.6 indicates the groups that participated in this study. The second column indicates the notations used by the participants to create the assurance case model. "N" in the third column indicates the number of participants involved in the study (68 participants in total from Group A and B who involved in the assurance case model construction task using SACM notation, and 69 participants who used GSN). The "Mean" column indicates the participants' average time to create an assurance case using a particular notation (presented in seconds). The "Variance" column indicates the variance from the gathered data of the participants' score in using each notation. The column "P-value" indicates the P-value associated with the mean generated from the participants' score using each notation. The "Alpha" column indicates the alpha number (margin of error), in this case, is 0.05.

As shown in Table 5.6, the average time spent by all the participants from both groups to create an assurance case model using SACM notation is 641.62 seconds and 647.72 seconds using the GSN. In general, this result indicates that the participants slightly quicker in creating an assurance case using SACM notation than using the GSN. The difference was only 6.10 seconds. Based on this result, we conducted a statistical test (using t-Test) to investigate whether we can statistically accept the Alternative Hypothesis (as the expected hypothesis) that indicates the average time required by the participants to create an assurance case model using SACM notation is less than the average time required by the participants to create an assurance case using GSN.

As shown in Table 5.6, the result of the test shows that the P-value is greater than the Alpha number, 0.05, that we used in this test. In this case, P-value is equal to 0.43, which means it is in the rejection region. Therefore, we need to admit although the average time required by the participants to create an assurance case using SACM notation is less than using GSN, the statistical test shows that this result is not statistically significant. We may conclude that we can not statistically accept the Alternative Hypothesis.

5.5.5 Identification of element variants

In this part, the result of the study related to the following question is described:

How effective is the produced SACM notation in the sense of its intuitiveness related to the identification of the correct variants of the element by the participant?

To answer the above question, we analysed the result from the survey questions given to the participants after completing each assurance case model construction task.

All the participants from Group A and B were asked to answer several survey questions related to identifying variants of a particular element of a notation. The questions were designed to study the notation's intuitiveness in the sense of identifying similar elements based on similar visual appearances.

There were 49 participants in Group A. They were asked to identify the variants of the SACM notation element first and then continue to identify the GSN element's variants. Meanwhile, there were 33 participants in Group B. They were asked to identify the variants of the GSN element first and then continue to identify the SACM notation element's variants. The participants were allowed to choose more than one answers.

Based on the participants' responses, we identified the effectiveness of the SACM notation in the sense of the visual representation intuitiveness in the context of the identification of related elements based on their similar visual appearances is considered low (less than 50% from total participants who can correctly identify the variant of the element) for the following elements:

- 'Claim' and its variants when provided by the information related to the 'asserted Claim'.
- 'Assumed' elements, in the case of 'assumed Context' and 'assumed Inference' relationship when being provided by the information related to the 'assumed Claim'.
- 'Axiomatic' element, especially the 'axiomatic Claim' when provided by the information related to the 'axiomatic Inference' relationship.
- 'Defeated' elements, in the case of 'defeated Inference' and 'defeated Context' when provided by the information related to the 'defeated Claim'.
- 'Abstract' elements, especially for the 'abstract Inference' and 'abstract Context' relationship when provided by the information related to the 'abstract Claim'.
- 'AsCited' elements, in the case of 'asCited Inference' and 'asCited Context' relationship when being provided by the information related to the 'asCited Claim'.

In general, we might say that the participants seem struggle to identify the similarity between the visual representation of the 'Claim' element and the relationship elements. This is indicated by the number of the correct answer from the participants were relatively low, for example, if they were provided by the information of a specific type of 'Claim' element and were asked to identify the related elements specifically the relationship-type elements. This finding can be used to improve the visual representation intuitiveness of the SACM notation, specifically related to the similarity aspects between elements.

Several elements received better result (more than 50% from total participants from both groups) in terms of the intuitiveness in the context of related elements identification based on their visual representation similarities:

- 'ArgumentPackage' elements. More than 50% from total participants were correctly identified the variant of 'ArgumentPackage' element.

- ‘Axiomatic’ element, specifically the ‘axiomatic Context’ relationship when provided by the information of the ‘axiomatic Inference’ relationship.
- ‘Abstract’ element, specifically the ‘abstract ArtifactReference’ when provided by the ‘abstract Claim’ information.
- ‘NeedsSupport’ element, specifically the ‘needsSupport Context’ relationship when provided by the information of the ‘needsSupport Inference’ relationship.

There is a tendency the participants relatively easy to identify the similarity between the visual representations that are designed using similar shape type such as in the context of the visual representations that are designed using the line-type shape.

The participants were also relatively easy to identify the related elements based on their *almost* identical visual representations such as in the case of ‘ArgumentPackage’ types. This argument is supported by the result found in the context of identifying the variant of GSN Module. Most of the participants (65% from total participants) were correctly identify the GSN Contract as the variant of the GSN Module. The design of the visual representation of the GSN Module and Contract are also almost identical. It is important to note that almost identical in this context can be defined as using the same shape types and location to form a visual representation of an element.

The result of the participants’ responses regarding the identification of each element variants can be found Appendix B.

5.5.6 Utilisation of visual variables

In the design of a notation, especially in SACM notation, visual variables were used as the building blocks to create the elements’ visual representation. The intuitiveness of several visual variables used to create the visual representation was assessed as an extra part during the experiment.

The visual variable Shape, especially used to represent the relationship between (node) elements, was evaluated. Several questions were designed and asked the participants to evaluate the intuitiveness the relationship visual representation that was designed using visual variable such as Shape. The intuitiveness of the relationship visual representation was considered necessary to be evaluated by the participants due to gather feedback from the participants related to the relationship element’s arrow direction to represent the relationship elements in the SACM notation.

The intuitiveness of the use of Size visual variable was also considered necessary to be evaluated. This is due to gathered feedback from the participants, especially in line size thickness for creating an ‘axiomatic’ visual representation of an element. The intuitiveness of the utilisation of the Brightness visual variable was also evaluated due to consider the application of it in the context of ‘counter’ element visual representation.

There are six questions in total asked to the participants:

- Four questions related to the utilisation of Shape (especially related to the use of arrow direction) and Position (for diagram layout):
 - Identifying the conclusion Claim when two Claims connected each other *vertically* via a relationship (line) *without* any decoration to indicates the source or target element.
 - Identifying the conclusion Claim when two Claims connected each other *vertically* via a relationship (line) *with* a dark brightness arrowhead used to indicates the target element.
 - Identifying the conclusion Claim when two Claims connected each other *horizontally* via a relationship (line) *without* any decoration to indicates the source or target element.
 - Identifying the conclusion Claim when two Claims connected each other *horizontally* via a relationship (line) *with* a dark brightness arrowhead used to indicates the target element.
- A question related to the utilisation of Size in the context of line thickness. In this case, the participants were asked to identify the conclusion Claim when two Claims (A and B) connected each other using a relationship (line) without any decoration to indicates the source or target element. In this case, both Claim A and B were drawn as a circle; however, with different size of line thickness; the circle of Claim A was drawn thicker than Claim B.
- A question related to the role of Brightness. In this case, the participants were provided by two different diagrams; the first diagram shows two Claims (A and B) connected each other vertically (Claim A above Claim B) via a relationship (line) with a *dark brightness* circle-head pointed to Claim A that is used to indicates the target element of the relationship. The participants were told the meaning of the first diagram that Claim A is supported by Claim B. The second diagram is relatively similar to the first diagram; the only difference is only the circle-head-line's brightness; in this case, it is drawn as *light brightness*. The participants were asked to identify the meaning of the light brightness circle-head-line in the diagram after they were given the information about the dark brightness circle-head-line.

The details of the above questions can be read further in Appendix B.6.

Seventy-one participants (40 from Group A and 31 from Group B) were involved in this task. For each question, the participants were allowed to chose exactly one answer from the given answer options. They were allowed not to answer a particular question if they considered no suitable answer options. The participants were also made aware that the diagram example presented in each question is not referring to a particular notation. This is due to reduce bias related to each question's participants' responses considering they participated in this task after learning both SACM notation and GSN.

Based on the participants' responses related to the arrowhead's role, we may conclude that a decoration such as an arrowhead can be used to help participants decide which element as the targeted element. The result (from the second and third questions) also confirmed that the layout of the elements in terms of their relative positions might not be affecting the participants' responses regarding the target element in the case if there was a decoration such as an arrowhead is used to indicate the target element.

The participants' responses from the task related to the role of shape in the context of the intuitiveness of a relationship visual representation, provide support to the design created for the direction of the relationship in SACM notation. The relationship visual representations in SACM notation were designed using a particular type of decoration to indicate the target element used in 'AssertedInference', 'AssertedEvidence', and 'AssertedContext' relationship.

Regarding the role of the Size visual variable in the context of line thickness, the participants' responses' shows that line thickness could provide "emphasise" to a particular element, for example, to indicates a "more" important" element relative to the others. This result provides support to the design of the 'axiomatic' element such as 'axiomatic Claim' that is designed using a thick line as part of the visual representation to indicates an emphasise that this element is declared to be axiomatically true and no further argumentation (elements) needed to support this element.

As for the role of Brightness visual variables especially in the context of to indicates a particular semantic in the relationship visual representation, the participants' responses shows that light brightness when being used as part of the relationship visual representation might indicate the meaning of "context" even though they were informed that the dark brightness is used to indicates "support"-type of relationship. We assumed that the learning effect, such as the effect after learning the GSN (InContextOf) relationship, might affect the participants' responses regarding this question.

The result of the participants' responses regarding the utilisation of visual variables can be found in Appendix B.

5.5.7 Participants' perception towards the notations

In this part, the participants' perception after learning and using the notation are described. There were three questions asked to the participants in this part:

1. Which notation do you considered as the notation that is easy to learn?
2. Which notation do you considered as the notation that is easy to use?
3. Which notation do you intend to use in practice when being asked to construct an assurance case?

There were 71 participants in total, 40 participants from Group A and 31 participants from Group B, who submitted their responses regarding the experience in learning and using both SACM notation and GSN.

Table 5.7 presents the summary of participants' responses from each group regarding their perceptions towards the experience in learning and using both SACM notation and GSN.

TABLE 5.7: Summary of participants' perception from each group: SACM notation and GSN

Group	Ease of learning	Ease of use	Intention to use
A (40 participants)	SACM (40%)	GSN (63%)	GSN (40%)
B (31 participants)	GSN (52%)	GSN (58%)	GSN/SACM (45%)

Based on the participants' responses as presented in Table 5.7, most of the participants in Group A (40%) chose SACM notation as the notation that is easy to learn, GSN as the notation that is easy to use (63%), and GSN as the notation that they intend to use in the future (40%). For the participants in Group B, most of them believed GSN as the notation that is both easy to learn (52%) and easy to use (63%). Surprisingly most of them also chose SACM notation as the notation they intend to use in the future beside GSN (45%).

The result presented on Table 5.7 indicates if the participants were given a chance to learn and use both SACM notation and GSN to create an assurance case model, there is a tendency that they intend to use GSN as the notation that they are willing to adopt in the future, even though, some participants, for example, from Group A, believed the SACM notation was relatively easy to learn than GSN.

To summarise, from total 71 participants (Group A and B) who submitted their responses regarding their perception towards the notations that they have learnt and used (as shown in Table 5.8), we may conclude, most of the participants believed that GSN as the notation that is easy to learn and use to construct an assurance case. Most of them also intend to adopt GSN as the notation that they are willing to use in the future when facing a similar problem (i.e., creating an assurance case model).

TABLE 5.8: Summary of participants' perception: SACM notation and GSN

Group	Ease of learning	Ease of use	Intention to use
A+B (71 participants)	GSN (38%)	GSN (61%)	GSN (42%)

The result of the participants' responses regarding their perception towards the notations can be found in Appendix B.

5.6 Threats to validity

Internal validity. The participants involved in the study were gathered from two different universities and majors (Computer Science and Information Technology). Their background knowledge might

influence the result of the conducted study. This is one of the most common threat for internal validity for the study involving the human subject.

Other than participants background knowledge, fatigue, or maturation threats might also influence the result of the study. To be noted, the study was conducted for approximately 50 minutes for each group with several tasks for them to participate. There could be a chance that some participants might lose focus on performing a particular task due to the duration and multiple tasks to be completed during the study.

Construct validity. In order to reduce participants' response bias, the participants were not being told until the end of the experiment that the SACM notation is the notation that is being monitored and the GSN is the comparative notation.

The study also followed that commonly-used 2×2 fractional factorial design in order to reduce the learning effects that might influence the participants' responses.

External validity. The result gathered from this study was in the context of 82 students as a sample from two different universities. A different result might be retrieved if the sample of students is different (different size and different university).

It is also unsafe to generalise the result of this study to industry professional context due to many different factors such as the background of knowledge and experience.

Conclusion validity. One general threat to conclusion validity is related to the number of samples of the experiment, which may reduce the ability to reveal patterns in the data. Eighty-two participants involved in the study, more participants, might reveal other or different patterns that could be resulting in more insight for the conclusion.

5.7 Challenges and limitations

There are challenges and limitations were identified related to the evaluation of the proposed visual notation design approach through an experimental study by comparing the effectiveness of the SACM notation as the notation that is developed based on the proposed approach and the GSN as an existing assurance case notation.

Study participants recruitment. At least 150 participants were invited to be involved in the experimental study from different universities in Indonesia. The invitation was delivered and informed to the participants approximately three weeks before the experiment with the help of the representative (academic staff) from both universities—however, only 82 participants who take part on the study.

Experiment environment. The evaluation of the proposed visual notation design approach was conducted through the assessment of the effectiveness of the SACM notation compared to the GSN in the context of the university students in Indonesia. Different participants (with a different background) might result in different evaluation result.

Time availability and experiment duration. The experiment was conducted for approximately 50 minutes for the participants in each group to involve. There were participants who could not get involved in all the tasks given during the experiment. This is due to time availability conflicting for several participants with other activities.

5.8 Summary and conclusion

The proposed visual notation design approach can be used to create an assurance case notation. This was demonstrated by the development of the SACM notation that has been discussed in Chapter 3. The effectiveness of the proposed visual notation design approach was evaluated through an experimental study involving 82 university students as participants by the evaluation of the effectiveness of the SACM notation, as the notation that is designed by following the steps of the proposed visual notation design approach, compared with an existing notation (GSN) that was developed using another approach.

The *effectiveness* of a notation, with accuracy, easy to learn, and easy to used as the characteristics of the notation, is used as a parameter to understand the effect of the application of the proposed visual notation design approach in designing an assurance case notation.

The following research questions related to the study were formulated and answered based on the result of the study:

1. *Which notation that is more effective in terms of its accurate used by the user to construct an assurance case model?*

Relative to the SACM notation, the GSN was found as the notation that was more effective in terms of its accurate used by the participants to create an assurance case model. The result related to this question was confirmed by a statistical test. Several SACM notation elements required further improvements in terms of its visual representation such as ‘ArtifactReference’ and types of ‘AssertedRelationship’.

2. *Which notation that requires less effort in terms of time required by the participants to create an assurance case model correctly?*

Relative to the GSN as the comparative notation in this study, the SACM notation was found as the notation that was more effective in terms of the time required by the participants to create

an assurance case model. The margin of the average time spent by the participants to create the model using the SACM notation and GSN is relatively small (only 6.10 seconds); hence, the result of the statistical test to confirmed the result indicates that it is not statistically significant.

3. *How effective is the produced SACM notation in the sense of the intuitiveness of the visual appearances of the element?*

Based on the visual representation description identification task, an improvement is required, especially for the elements that were designed using icon-type visual representation. We identified less than 50% participants who can identify the correct meaning of the visual representation of the SACM notation, in this case, is the visual representation of the ‘ArtifactReference’ and the ‘ArgumentPackage’.

4. *How effective is the produced SACM notation in the sense of its intuitiveness related to the identification of the correct variants of the element by the participant?*

Based on the element variants identification task, the SACM notation was considered effective in terms of the design similarity that can help the user to identify related elements such as in the case of ‘ArgumentPackage’ and its variants, ‘axiomatic’ element variants, ‘abstract’ element variants, and ‘needsSupport’ element variants. An improvement is considered necessary for several elements such as ‘Claim’ and its variants, ‘assumed’ element variants, ‘axiomatic’ element variants, and ‘defeated’ element variants.

5. *Which notation that the participants’ considered is relatively easy to learn?*

Most of the participants (38%) considered the GSN as the notation that was easy to learn during the experiment. As for the SACM notation, there were 31% who believed that the SACM notation was relatively easy to learn. There were also 24% of participants who considered both notations were easy to learn, and 4% of the participants stated that both of the notations were not easy to learn. The rest of the participants chose not to declare their perception regarding this question.

6. *Which notation that the participants’ considered is relatively easy to use?*

Most of the participants (61%) were considered the GSN as the notation that was easy to use to create an assurance case model. Only 18% participants who believed that the SACM notation that is relatively easy to use to create an assurance case model. There were also 10% participants who believed that both notations were relatively easy to use, and 8% participants considered both notations were not easy to use. There were also 3% participants who chose not to declare their perception.

7. *Which notation that the participant intends to use in the future when facing similar problems (create an assurance case)?*

Based on the participants' perception, 42% participants intend to use the GSN in the future to create an assurance case mode. As for the SACM notation, there were 34% participants who intend to adopt SACM notation. There were also 18% participants who have the intention to adopt both notations and 3% participants who chose not to adopt any of the notations. There were also 3% participants who chose not to declare their perception.

As part of the experimental study, survey questions regarding utilising visual variables in notation design were also asked the participants, especially related to utilising the Shape, Size, and Brightness visual variable relationship-type visual representation design. In general, the result shows that the design of relationship visual representation in SACM notation is relatively intuitive in the context of similar to the participants' perception to indicates, for example, the target element when being used in a diagram.

To summarise, the proposed visual notation design approach can be used to create a visual notation based on a defined metamodel. This is shown by the development of the SACM notation. However, based on the evaluation involving novice users discussed previously in this chapter, the resulting SACM notation is considered needed an improvement in terms of its effectiveness when compared to an existing notation (GSN). The design decision made during the development of the SACM notation might not be acceptable so that it might influence the effectiveness of the notation. We conclude that the proposed visual notation design approach can be used to create a notation based on a metamodel; however, the design decisions taken as part of the proposed approach might affect the effectiveness of the notation.

In the next chapter, the evaluation regarding the design decision made during the development of the SACM notation is discussed.

Chapter 6

Evaluation of the SACM notation

6.1 Introduction

In the previous chapter, the evaluation of the proposed visual notation design approach has been discussed. The evaluation was conducted by assessing the effectiveness of the SACM notation, as the notation that is designed based on the proposed approach, relative to an existing notation, the GSN, that is designed based on another approach.

In this chapter, the evaluation focuses on the following aspects:

- Evaluation of the effect of the design decisions made during the development of SACM notation.

This evaluation aims to investigate the effectiveness of the design decisions that were made during the development of the SACM notation. With this objective, an alternate SACM notation was created by following the same approach - the proposed visual notation design approach - however, the design decisions taken during the creation of the visual representation were made different with the SACM notation. An experimental study was designed to be similar to the experimental study discussed in Chapter 5 to evaluate the effectiveness of both SACM notations and its alternate notation.

- Experienced user evaluation.

This evaluation aims to seek feedback from experienced assurance case stakeholders regarding the produced SACM notation relative to their knowledge and experience related to the assurance case notation that they regularly used. A survey study was designed and conducted to achieve this objective. The result of the study will be considered as one of the inputs for the further development of SACM notation.

- Evaluation through tool support.

The objective of this evaluation is to show that the proposed notation can be applied in tool support. The application of the SACM notation in tool support might increase the adoption of SACM notation by the users.

The detail of each evaluation part of the SACM notation is discussed in the next sections in this chapter.

6.2 Design decision evaluation

6.2.1 Study definition

6.2.1.1 Objective

The proposed visual notation design approach consists of four steps, as previously discussed in Chapter 3. In the first three steps, the notation designer needs to make decisions regarding the development of the notation. For example, in step 1 (create a design path), the notation designer is encouraged to decide the design order to create the visual representation of the identified elements. In step 2 (consider design constraints), the notation designer needs to decide, for example, which existing notations and which existing design principles that are considered as part of the design constraints in the development of the notation. In Step 3 (create visual representation), the notation designer needs to decide the composition of visual variables that are used to create the visual representation of the elements.

In Chapter 4, the application of the proposed visual notation design approach has been demonstrated in the development of SACM (argumentation) notation. Design decisions were made as part of the development of the SACM notation. In this section, the evaluation of the design decision, especially related to the design decision in Step 3 of the development of SACM notation is discussed. The utilisation and composition of visual variables to create the visual representation of the elements may affect the effectiveness of the notation [3, 6, 54]. By the evaluation of the design decision that was made related to utilisation and composition of the visual variables for the creation of the SACM notation, the result can be used to provides an understanding regarding how effective is the design decision that was made for the creation of SACM notation. The evaluation result is also expected to be used as one of the inputs for further improvement of the SACM notation.

In order to understand the effectiveness of the design decision of the SACM notation, an experimental study was designed and conducted by comparing the SACM notation and its alternate notation. An alternative SACM notation was created by following the same visual notation design approach as used to develop the SACM notation; however, the design decisions made related to the utilisation and composition of the visual variables were different with the decisions made for the development of the

SACM notation. This is due to investigate the effect of the design decision on the effectiveness of the notation.

The experimental study for the evaluation of the design decision effectiveness of SACM notation that is discussed in this section was designed to be similar to the experimental study that was conducted to evaluate the effectiveness of the SACM notation relative to an existing notation that is developed by using another design approach in Chapter 5. The parameter used to evaluate the notation was also similar as discussed in Chapter 5: the effectiveness of the notation in the context of accurate use of the notation by the user, easy to learn, and easy to use as the monitored characteristics. The novice notation user were considered as the participants for this study due to gather objective feedback, in this context, the university students with no prior knowledge and experience related to the assurance case notation were selected.

To summarise, we state the objective statement in the form prescribed by Wohlin et al. in [92] as the following:

Analyse the SACM notation and its alternate notation.

For the purpose of evaluating the design decisions of the SACM notation

With respect to the effectiveness of the notation.

From the point of view of the University students.

In the context of assurance case model experiment (construction task and notation intuitiveness studies).

6.2.1.2 Experimental objects

The experimental objects of this study were:

- The SACM notation. It is developed by using the proposed visual notation design approach.
- The alternate notation of SACM. It is developed by using the same approach with the SACM notation; however, with different design decisions related to the utilisation and composition of the visual variables to create the visual representation of the elements.

For example, ‘Claim’ in the SACM alternate notation is represented using an ellipse due to considering an index visual representation to the existing notation (CAE) that represents a ‘Claim’ using an ellipse. The ‘AssertedRelationship’ is represented using a line with a text indicator placed in each line-end to indicate the source and target elements. The text "S" is used to indicate the source element, and to indicate the target element, the text "T" is used. A single

line with solid texture represents the ‘AssertedInference’ as the sub-class of the ‘AssertedRelationship’. A single line with a solid texture and a half circle as a decoration placed at the source line-end is used to represent the ‘AssertedEvidence’ relationship. A double-lines with solid texture is used to represent the ‘AssertedContext’. The visual representation of other ‘AssertedArtifactSupport’ is created to be identical with the visual representation of the ‘AssertedEvidence’, and the visual representation of the ‘AssertedArtifactContext’ is created to be identical with the visual representation of the ‘AssertedContext’. The use of the same visual representation to represent different elements such as used in ‘AssertedArtifactSupport’ and ‘AssertedArtifactContext’ due to consider graphic complexity in a notation [3]. The use of ‘AssertedEvidence’ and the ‘AssertedArtifactSupport’ can be differentiated by identifying the source and target elements attached to the relationship. Similarly, the use of ‘AssertedContext’ and ‘AssertedArtifactContext’ can be differentiated by identifying the source and target elements attached to the relationship. The source and target elements of the ‘AssertedArtifactSupport’ ‘AssertedArtifactContext’ must be a type of ‘ArtifactReference’.

The ‘Claim’ and each type of ‘AssertedRelationship’ can be defined into a specific type, for example, based on the enumeration literals of the ‘AssertionDeclaration’. To accommodate this, in terms of visual representation, each type of declaration visual representation is made to be consistent to be applied for ‘Claim’ and each type of ‘AssertedRelationship’ element. A square with a specific type of decoration can be attached within (centre-top of) the ellipse. When applied in an ‘AssertedRelationship’ type, the square must be placed in the middle of the line. To indicate an element that has been ‘Defeated’, a cross symbol is adopted to indicate a crossing-out, i.e., the element has been defeated by counter-evidence/argument. To indicate an assumed element, an exclamation mark (!) is placed within the square to indicate that the user needs to be careful with this element since it is declared as an assumed element. The other example of visual representations of ‘Claim’ and ‘AssertedRelationship’ type can be seen in Figure 6.1. Generally, they were made using different design decision with the SACM notation; however, the design approach used is similar to the design approach used in SACM notation, including the visual inheritance mechanism.

As shown in Figure 6.1, each type of ‘Claim’ and ‘AssertedRelationship’ used the same and consistent decoration to indicate a specific declaration definition. For example, a square within a box indicating ‘defeated’ is consistently used to represent ‘defeatedClaim’ and ‘defeatedInference’. This is due to respecting the notation design approach applied in the development of the SACM alternate notation.

Another critical different design decision related to the SACM alternate notation is about the elements’ layout when combined in a diagram. The SACM notation generally drawn using a top-down approach where the main ‘Claim’ is located on the top-level of the diagram supported by other elements below the main ‘Claim’ and for the contextual elements are placed

Asserted Element	ID statement Claim	<u>T</u> ID <u>S</u> AssertedInference	<u>T</u> ID <u>S</u> ₁ AssertedEvidence AssertedArtifactCont	<u>T</u> ID <u>S</u> AssertedContext AssertedArtifactContext		
Declaration type	☒ Defeated	☐? NeedsSupport	☐# Abstract	☐! Assumed	☑ Axiomatic	☐(y) AsCited
Example: Defeated element	ID ☒ statement defeatedClaim	<u>T</u> ID <u>S</u> defeatedInference	<u>T</u> ID <u>S</u> ₁ defeatedEvidence defeatedArtifactConte	<u>T</u> ID <u>S</u> defeatedContext defeatedArtifactContext		

FIGURE 6.1: SACM Alternate notation (Example of ‘Claim’ and ‘AssertedRelationship’ type)

horizontally relative to the target element. In SACM alternate notation, the layout is different. In general, the elements needs to be drawn horizontally from left to right. The main ‘Claim’ needs to be drawn thicker and bigger than other elements (to indicate it is the main ‘Claim’) and the supporting elements must be placed on the right side of the main ‘Claim’. The contextual elements need to be drawn vertically relative to the target element. Figure 6.2 illustrates an example of an assurance case drawn using the SACM alternate notation.

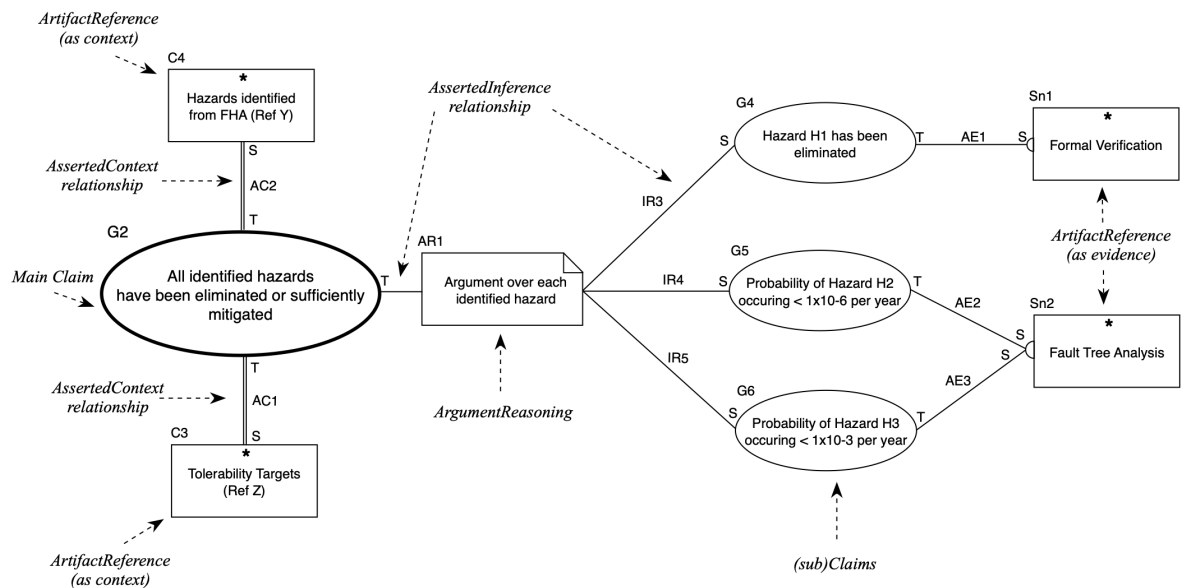


FIGURE 6.2: SACM Alternate notation example

The summary of the created visual representation for the SACM alternate notation elements is available in Appendix C.

6.2.1.3 Perspective

The experiment was conducted from the perspective of University students majoring in Computer Science and Information Technology from Telkom University in Indonesia with no prior knowledge and experience related to the assurance case notations.

6.2.2 Planning

Similar research questions were considered in this study as used in the evaluation study as discussed in Chapter 5, however, the Alternate SACM notation is used as the comparative notation in this study.

1. Which one that is more effective in terms of its accurate used by the participant to construct an assurance case model?
2. Which notation that requires less effort in terms of time required by the participant to create an assurance case model?
3. How effective is the produced SACM notation in the sense of the intuitiveness of the visual appearances of the element?
4. How effective is the produced SACM notation in the sense of its intuitiveness related to the identification of the correct variants of the element by the participant?
5. Which notation that the participants' considered is relatively easy to learn?
6. Which notation that the participants' considered is relatively easy to use?
7. Which notation that the participant intends to use in the future when facing similar problems (create an assurance case)?

6.2.2.1 Participant selection criteria

The participants were selected through a non-probabilistic convenience sampling approach. In this study, the invited participants were the students from Telkom University (Indonesia) majoring in Computer Science and Information Technology.

6.2.2.2 Context selection

The university students in this experiment were referred to as "participants". This experiment was conducted by the support of the Telkom University. An academic staff as the representative from the university help circulating the information regarding the experiment. The participants were informed and invited to the study by the help of the representative from the university for approximately 3

weeks before the experiment was conducted. A classroom equipped with PCs for the participants to be used in this experiment was also provided by the university.

6.2.2.3 Hypothesis formulation

The following hypothesis was formulated to compare the effectiveness of the SACM notation and its alternate notation in the context of the accurate use of the notation by the participant to create an Assurance Case model ('AC'):

- $H0_{AC}: AC_{Alt} \Rightarrow AC_{SACM}$
- $H1_{AC}: AC_{Alt} < AC_{SACM}$

The ' AC_{SACM} ' represent the score (average correct answer) in using the correct visual representation to represent the SACM element in an assurance case model construction task. The ' AC_{Alt} ' represent the score in using the correct visual representation to represent the alternate SACM notation element in an assurance case model construction task. 'H0' is the null hypothesis, which is true when the average score from total participants using the Alternate SACM notation to create an assurance case model is equal or better than using the SACM notation. 'H1' is the alternative hypothesis (expected), which is true when the average score from total participants using the SACM notation to create an assurance case model is better than using the alternate SACM notation.

The following hypothesis was formulated to compare the easy to use characteristics of the SACM notation and its alternate notation in terms of Time Spent ('TS') to create an assurance case model using a particular notation:

- $H0_{TS}: TS_{Alt} \leq TS_{SACM}$
- $H1_{TS}: TS_{Alt} > TS_{SACM}$

' TS_{Alt} ' represents the average time spent by the participant to create an assurance case model using alternate SACM notation. ' TS_{SACM} ' represents the average time spent by the participant to create an assurance case model from the same case study using SACM notation. 'H0' is the null hypothesis, which is true when the average time required to create an assurance case model using SACM notation is greater than or equal to the average time required to create an assurance case model using Alternate SACM notation. 'H1' is the alternative hypothesis (expected), which is true when the average time spent by the participants to create an assurance case model using SACM notation is less than the average time spent by the participants to create an assurance case model using the Alternate SACM notation.

6.2.2.4 Variable selections

Dependent variables: the effectiveness of the notation in the sense of the accurate use and easy to use characteristics of the notation by its users.

Independent variables: the SACM notation and the alternate SACM notation

Environmental variables: the learning speed of the participants.

6.2.2.5 Study design

Table 6.1 shows the planned design for the experimental study. In general, the design of the study is relatively similar with the design of the study as discussed in Chapter 5. The difference only related to the notation that is used as the comparative notation, in this case, the alternate SACM notation is used as the comparative notation.

TABLE 6.1: Study design: SACM notation and its alternate notation

Group A	Group B
Introduction and study information	
Ethics and data privacy explanation (consent form)	
Demographics	
Part 1: Visual representation intuitiveness	
Part 2: 2.1. Read and received basic SACM notation tutorial 2.2. Create an assurance case model using SACM notation 2.3. Identify SACM element variants 2.4. Read and received basic Alternate SACM notation tutorial 2.5. Create an assurance case model using Alternate SACM notation 2.6. Identify Alternate SACM notation element variants	Part 2: 2.1. Read and received basic Alternate SACM notation tutorial 2.2. Create an assurance case model using Alternate SACM notation 2.3. Identify Alternate SACM notation element variants 2.4. Read and received basic SACM tutorial 2.5. Create an assurance case model using SACM 2.6. Identify SACM element variants
Part 3: Visual variables utilisation in a notation	
Part 4: Participant's perception towards the notations	
Participants' feedback	

6.2.2.6 Instrumentation

The following materials were provided to the participants during the study:

- A printed copy of the SACM notation documentation (basic elements).
- A printed copy of the Alternate SACM notation documentation (basic elements).
- A printed copy of case study instruction. The same case study as used in the study as discussed in Chapter 5.
- Questions related to the experiment that was provided in an online survey platform. The order of the questions was designed differently for two different groups in this experiment. Therefore, each group was provided by two different address to access the online survey platform.

6.2.3 Operation

6.2.3.1 Preparation

All the experiments materials were prepared before the experiment was conducted. A pilot study using the exact study design was conducted before the main experiment was executed. The objective was to identify unanticipated obstacles and constructing a simulation of the main experiment. The result from the pilot shows no significant error was identified. A refinement was made to the case study documentation mostly related to the typo in writing.

6.2.3.2 Execution

The execution of the main experiment is relatively according to plan. The participants involved in the experiment based on the tasks given as planned in the study design. The participants were divided into two groups and following the instruction for each part of the experiment such as start from reading the study objective until the last task - providing feedback related to the notation and the study. The experiment was conducted for approximately 50 minutes for each group, and there was 2 experiment assistant who helps to make sure the experiment run as the study plan.

6.2.3.3 Data validation

All the participants who involved in the pilot study were never had prior knowledge related to the assurance case notation. This is confirmed by their responses regarding the question related to their experience in assurance case notation. The feedback from the participants from the pilot study was considered as an input to improve the preparation for the main study. These participants were also not allowed to participate in the main study in order to avoid bias responses.

As for the participants from the main study, they were confirmed to be valid to participate in the study considering their responses regarding the question related to their experience in assurance case notation. Based on this, we considered the data gathered for this study are valid and relevant to the study objective.

6.2.3.4 Data collection

An online platform was used to collect the participants' responses data for this study. The participants' from Group A were asked to access the following link to participate in the study:

<https://selviandro.limequery.com/328938?lang=en>

The following link was provided for the participants' in Group B:

<https://selviandro.limequery.com/738511?lang=en>

The only data that was gathered manually is related to the assurance case model construction task, including the timing data to complete the model.

The data were collected from participants from Telkom University, Indonesia. In total, there were 60 participants involved in this experiment: 25 participants in Group A and 35 participants in Group B. The experiment was conducted on the 27 November 2019 in the morning for Group A, and in the afternoon for Group B. The experiment was conducted in a classroom equipped with PCs with internet connection for the participants to use during the experiment.

6.2.4 Data analysis and interpretation

6.2.4.1 Demographics

There were several demographics questions asked to the participants related to their background and experience.

The first question was related to the participant's domain of work. Several answer options were provided for the participant to choose. They may also type their answer for this question as part of the "Other" answer option. The participants may choose more than one answer that relevant to their condition.

From a total of 60 participants, all of them was a university student. There were 3 participants who also work as an "Academia" (e.g. Research Assistant), 1 participant who also works as a "Teaching Assistant", and 1 participant who also work as "System Admin".

The participants were asked about their experience related to the software/system modelling and notations in the second. 59 out of 60 participants claimed that they have experience related to the system/software modelling notations. Only one participant answered "No" for this question. As a follow-up question for the participants who have experience in system/software modelling notations, the participants were asked about the notation that they have experience with. Based on the participants' responses, most of them have experience with the ERD notation (59 participants), followed by BPMN (26 participants) and UML (25 participants).

The participants were asked about their experience and knowledge related to the assurance case modelling and notation in the following question. Of the 60 participants, all of them claimed that they never have any prior experience related to the assurance case modelling and notation. This result confirmed that the participants are relevant to participate in this study, and we considered their responses to the questions and tasks in this study to be valid.

6.2.4.2 Visual representation intuitiveness

In this part, the participants were asked two questions related to the intuitiveness of the SACM elements visual representation. The participants' responses were used to answer the following research question:

How effective is the produced SACM notation in the sense of the intuitiveness of the visual appearances of the element?

The first question asked to the participants was related to the intuitiveness of the 'ArtifactReference' visual representation. The participants were given the visual representation of the 'ArtifactReference', and they were asked to identify the most suitable description for that visual representation. Several answer options were provided for the participants to choose.

Based on the participants' responses (as illustrated in Figure 6.3), from total 60 participants, most of them (40 participants or 67% from total participants) were identified the correct description for the 'ArtifactReference' visual representation ("*a document/artifact icon that can be used as a reference to evidential or contextual information to support a structured argument*"). A small number of participants who choose other options were also identified such as "*a note that can be used to describe additional information*" (11 participants) and "*a package for containing structured arguments*" (9 participants). This result indicates that the 'ArtifactReference' visual representation was found relatively intuitive for this group of participants.

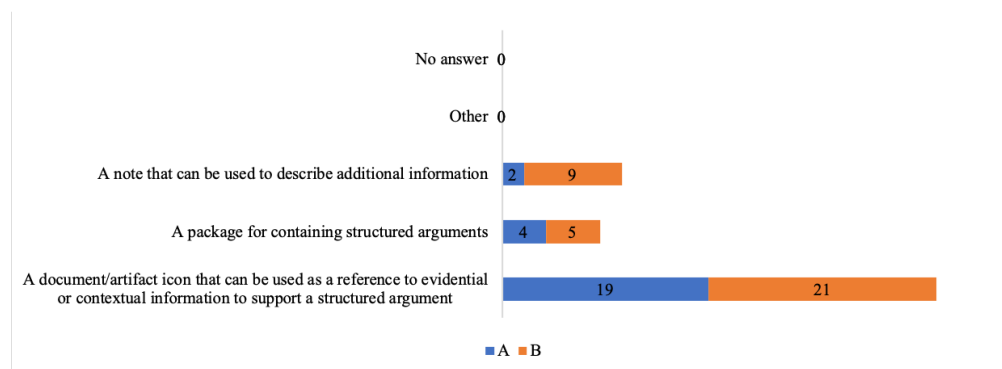


FIGURE 6.3: The participant's responses regarding the intuitiveness of the 'ArtifactReference' visual representation

The second question asked to the participants was related to the intuitiveness of the 'ArgumentPackage' visual representation. They were asked to identify the most suitable description for the 'ArgumentPackage' visual representation. Similar answer options as given in the first question were provided to the participants. The participant only allowed to choose one answer. They may type their own description of the 'ArgumentPackage' visual representation in case they could not find one based on the provided answer options. The participants' responses regarding the second question is illustrated in Figure 6.4.

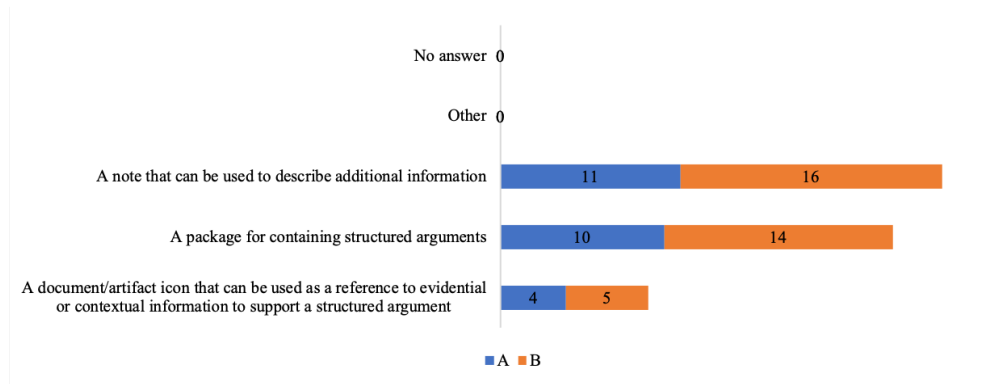


FIGURE 6.4: The participant's responses regarding the intuitiveness of the 'ArgumentPackage' visual representation

From total 60 participants, only 24 participants (40%) who can identified the correct description of the 'ArgumentPackage' visual representation ("*a package for containing structured arguments*"). There were 27 participants (45%) who chose "*a note that can be used to describe additional information*" and 9 participants (15%) chose "*a document/artifact icon that can be used as a reference to evidential or contextual information to support a structured argument*" as the most suitable description for the 'ArgumentPackage' visual representation. This result indicates that further work is needed in order to improve the intuitiveness of the 'ArgumentPackage' visual representation.

6.2.4.3 Accurate use of the notation

In this part, we described the result of the study related to the following question:

Which notation (between SACM notation and its alternate notation) that is more effective in terms of its accurate use by the participants to construct an assurance case model?

To answer the above question, we designed and conducted an assurance case construction task using both SACM notation and its alternate notation for all Group A and B participants.

Group A participants learned about the SACM notation first and used it to create an assurance case model based on the given case study. After that, they learned about the SACM alternate notation and used it to create the same case study.

Group B participants learned and used the SACM alternate notation first to create an assurance case model, and then learned used the SACM notation to create another model using the same case study.

Different treatment in terms of the sequence of the notation that the participants learned and used was design to understand whether the learning effect might affect the result of the study.

The following materials were provided to the participants:

- SACM notation (basic elements) documentation.
- SACM alternate notation (basic elements) documentation.
- Case study instruction.

In this study, we used the same case study used in the experiment discussed in Chapter 5. In this case, the participants were provided by a printed case study instruction that provides an assurance case about the safety aspect of a control system adopted from the GSN community standard. The adopted assurance case was presented in a tabular presentation (as shown in Figure 5.3).

We constructed the correct representation of the assurance case based on the case study using both SACM notation and SACM alternate notation, which can be seen in Appendix D as a benchmark to identify the element's correctness.

Suppose the assurance case model from the case study is constructed using SACM notation or SACM alternate notation. In that case, there must be 20 elements that must be drawn correctly by the participants in their model, with details as follow:

- 4 'Claim' elements (Claim G1, G3, G7, and G8)
- 1 'ArgumentReasoning' (S2)
- 6 'ArtifactReference' elements (4 as Context (C1, C2, C4, and C5), and 2 as Evidence (Sn3 and Sn4))
- 3 'AssertedInference' elements (relationship between G3 and G1, G7 and G3, G8 and G3)
- 2 'AssertedEvidence' elements (relationship between Sn3 and G7, Sn4 and G8)
- 4 'AssertedContext' elements (relationship between C1 and G1, C2 and G1, C4 and G3, C5 and G3)

For each created assurance case model, we calculated its score by identifying the correctness of each element. For each element, we give 1 point if the element's visual representation is drawn correctly and another 1 point if the element's positional placement is correct in the diagram. We then normalised the maximum score for each created model to 100. In the case of a model created using the SACM notation, the participant will get a 100 score if they correctly created the visual representation and the positioning of all 20 elements in the model. Similarly, for the model created using the SACM alternate notation, the participant will get 100 if they correctly created the visual representation and the positioning of all 20 elements in the model.

From the result of the assurance case model task, we compared the calculated average score of the assurance case models created using the SACM notation by all the participants from Group A and

B, with the average score of the assurance case models that were created using the SACM alternate notation by all the participants from both Groups.

Table 6.2 shows the calculated average score from both Groups in creating an assurance case model using SACM notation and SACM alternate notation.

TABLE 6.2: Average score of the assurance case construction task using SACM notation and SACM alternate notation

		Average Score	
		SACM notation	SACM alternate notation
Group A		74.27 (24 participants)	81.77 (24 participants)
Group B		63.36 (35 participants)	66.02 (32 participants)
Total average (Group A + B)		67.80 (59 participants)	72.77 (56 participants)

Based on data presented in Table 6.2, the total average score from Group A and B participants in creating assurance case model using SACM alternate notation is better than the average score of the Group A and B participants when creating assurance model using SACM notation. This result indicates that the design decision might influence the effectiveness of a notation in terms of its accurate use to create an assurance case model, considering the SACM notation and SACM alternate notation were created based on the same visual notation design approach, however, with a different design decision.

According to the data presented in Table 6.2, the Group A participants received a better score in creating an assurance case model using SACM alternate notation than using the SACM notation. This might be due to the learning effect since Group A participants learned and used the SACM notation first and then learned and used the SACM alternate notation. They might use their previous knowledge (learning and using SACM notation) to create another assurance case model using SACM alternate notation.

Group B participants received a better score in creating an assurance case model using SACM alternate notation as the first notation that they learned than using the SACM notation as the second notation that they learned. This indicates that learning order might not affect the performance of Group B participants.

Table 6.3 summarises the assurance case model construction task result based on Group A and B participants' performance.

As presented in Table 6.3, we identified nine participants (5 from Group A and 4 from Group B) who get a perfect score (100) in creating assurance case model using SACM notation. However, no participant got a perfect score in creating an assurance case model using SACM alternate notation.

TABLE 6.3: Assurance case model construction task result: group comparison

	Group A		Group B	
	SACM notation	SACM alternate notation	SACM alternate notation	SACM notation
Model received	24	24	32	35
Average score	74.27	81.77	66.01	63.35
Max score	100 (5 participants)	97.5 (8 participants)	97.5 (4 participants)	100 (4 participants)
Min score	17.5 (1 participant)	17.5 (1 participant)	20 (1 participant)	12.5 (1 participant)
Most accurately drawn element	Claim (79%)	AssertedContext (88%)	ArgumentReasoning (53%)	Claim (51%)
Most inaccurately drawn element	ArtifactReference (as Context: 46% as evidence: 42%)	Claim (0%)	Claim (19%)	ArtifactReference (as Context: 23% as evidence: 26%)

On average, both groups' participants tend to get a better score in creating an assurance case using SACM alternate notation than using SACM notation. However, none of the participants who get a perfect score when using the SACM alternate notation.

Based on the participants' performance in the assurance case model construction task, SACM notation 'Claim' was found as the type of element that most accurately drawn by Group A and B participants. 79% participants correctly drawn all 'Claim' elements' visual representation and its placement in the diagram. 'Claim' G1 (main 'Claim' in the model) was found to be the most accurately drawn by the participants from both groups, and 'Claim' G8 was found as the element that participants most struggled to draw.

Both Group A and B participants found the 'ArtifactReference' as the element that they most struggled to draw accurately, both in terms of correct visual representation and its placement in the diagram, especially when used as evidence for Group A participants, and when used as context for Group B participants. Common mistakes found, for example, the use of Rectangle to draw an 'ArtifactReference', no arrow found as part of the 'ArtifactReference' visual representation, and the use of diamond to indicates that the participants could not remember the 'ArtifactReference' visual representation.

As for the SACM alternate notation, 'AssertedContext' relationship was found as the element that most Group A participants accurately draw in their model and 'ArgumentReasoning' for the Group B participants. Group A and B participants found it struggled to draw the 'Claim' element. No Group A participant can accurately draw all 'Claim' elements in their model, and only 19% Group B participants can accurately draw all the 'Claim' elements. Most of the participants struggled to draw the 'Claim' G1 element using the SACM alternate notation. This is because the main 'Claim' in SACM alternate notation must be drawn bigger and thicker than other 'Claims' to indicates that it is the main 'Claim' in the model. This result, especially in this context, indicates that the use of visual variable size in terms of the size of a particular element and line size to draw the element might negatively impact the accurate draw of a particular notation.

The result regarding the other elements of SACM notation and SACM alternate notation can be read further in Appendix D.

Hypothesis testing

In this chapter, previously, we have formulated the hypothesis to evaluate the effectiveness of the SACM notation and SACM alternate notation in the context of the accurate use of the notation by the participants.

Statistical hypothesis testing using a Two-Sample t-Test (for unequal variances) was conducted to evaluate the effectiveness of the SACM notation and SACM alternate notation in the context of the participants' accurate use of the notation. The data distribution for this study is shown using a histogram as illustrated in Figure 6.5. The data distribution was found non-normally distributed, but it is not affecting the validity of the t-Test ([93]).

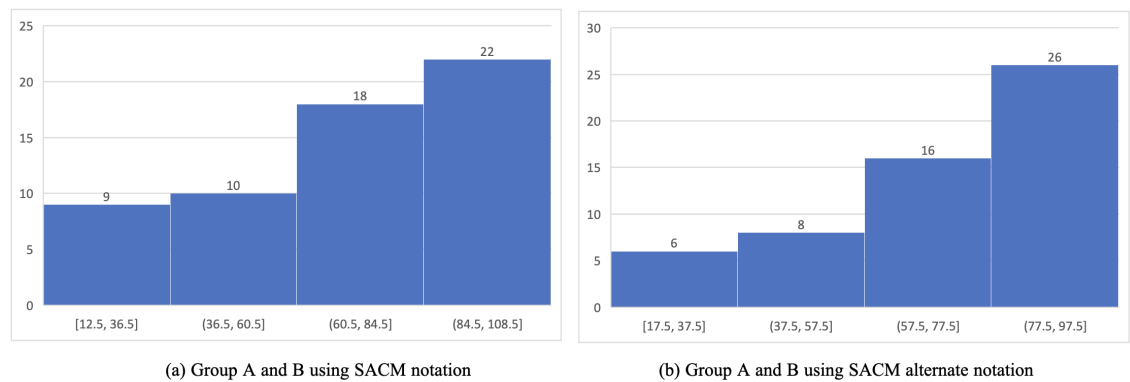


FIGURE 6.5: Data distribution Group A and B for the accurate use of the notation study (using (a) SACM notation and (b) SACM alternate notation)

The result of the test is presented in Table 6.4.

TABLE 6.4: t-test results for the accurate use of the notation study (SACM Vs Alternate notation)

Group	Notation	N	Mean	Variance	P value	Alpha
A+B	SACM	59	67.80	726.96	0.14	0.05
	SACM Alt	56	72.77	487.43		

The first column of Table 6.4 indicates the groups participating in this study. The second column indicates the notations used by the participants to create the assurance case model. "N" in the third column indicates the number of participants involved in the study. In this case, 59 participants in total (from Group A and B) were involved in the assurance case model construction task using SACM notation, and 56 participants who used SACM alternate notation. The "Mean" column indicates the resultant mean from participants' score in the assurance case model construction task. The "Variance" column indicates the variance from the gathered data of the participants' score in using each notation. Since the variance from the data using the SACM notation and SACM alternate notation is unequal, the Two-Sample t-Test for unequal variances was used to statistically evaluate the data. The column

"P-value" indicates the P-value associated with the mean generated from the participants' score using each notation. The "Alpha" column indicates the alpha number (margin of error), in this case, is 0.05.

As shown in Table 5.3, the assurance case model construction task's average score using SACM notation was 67.80 and 72.77 as the average score using the SACM alternate notation. This means the accurate use of the SACM alternate notation is better than SACM notation (in this case, the result indicates something that we are not expected). Therefore, we need to test whether the Null Hypothesis can be statistically accepted (Null Hypothesis stated that accurate use of SACM alternate notation is better than SACM notation). By conducting the t-Test, the resulting P-value is equal to 0.14, in this case, it is bigger than the Alpha number that we used in this test (0.05), and it is within the rejection region. Therefore, we may conclude that we cannot statistically accept that the SACM alternate notation is more effective than SACM notation in the context of accurate use of the notation by the participants based on the assurance case construction task.

6.2.4.4 Notation that is easy to use

In this part, the result of the study related to the following question is described:

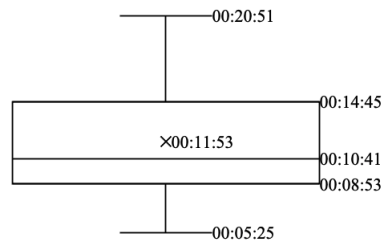
Which notation (between SACM notation and its alternate notation) that requires less effort in terms of time required by the participants to create an assurance case model correctly?

The data used to answer the above question is the time data that the participants spent to create an assurance case model by using a particular notation in the assurance case model construction task.

There were 59 models created by Group A and B participants using SACM notation. As presented in Figure 6.6, the participants' average time from both groups to create an assurance case using SACM notation was 11:53, with the longest time 20:51, and 05:25 for the shortest. As for the assurance case created by the Group A and B participants using SACM alternate notation, we received 56 models in total, with the average time 11:49, 16:24 for the longest, and 06:10 for the shortest. Based on this data, on average, the participants relatively quicker, creating an assurance case using SACM alternate notation than using SACM notation; the time difference is only 0.04 second.

We also calculated the average time taken by the participants who get a perfect score in creating the assurance case models using SACM notation and SACM alternate notation as summarised in Table 6.5. Since no participant found from both groups got a perfect score in creating an assurance case using SACM alternate notation, no average time can be calculated. Nine participants got a perfect score in creating an assurance case using SACM notation from both groups, with the average score 09:34.

Timing - SACM notation (Group A and B)



Timing - SACM alternate notation (Group A and B)

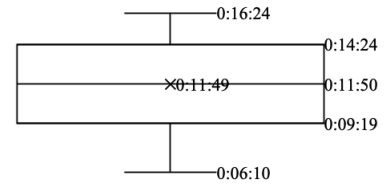


FIGURE 6.6: Performance results of Group A and B with respect to the times spent in creating the assurance case models using both notations (in hh:mm:ss)

TABLE 6.5: Timing data: average time spent by Group A and B participants to create the correct assurance case models using SACM notation and SACM alternate notation

	SACM notation	SACM alternate notation
Number of models with a perfect score	9	0
Average time spent	00:09:34	00:00:00

We also identified that the participants tend to spend more time creating the assurance case model using the first notation they learned during the experiment. After they have experience with the first notation, they tend to spend less time in constructing the model using the second notation that they just have learnt. This is indicated by the average time spent by the participants in Group A in creating the assurance case model using the SACM notation (as the notation that they learned first during the experiment) is greater than the average time spent by them in creating the models using SACM alternate notation (as the second notation that they learned during the experiment). As shown in Figure 6.7, the average time using SACM alternate notation 11:42 and the average time using SACM notation 15:20.

Timing - SACM notation Group A



Timing - SACM alternate notation Group A

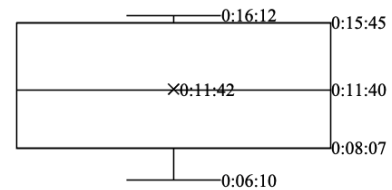
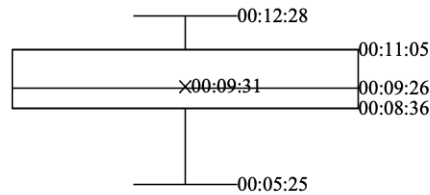


FIGURE 6.7: Performance results of Group A with respect to the times spent in creating the assurance case using SACM notation and SACM alternate notation(in hh:mm:ss)

A similar pattern was found in assurance case model construction by Group B participants, as can be seen in Figure 6.8, Group B participants relatively quicker in creating assurance case using SACM notation (as the second notation that they learned during the experiment) than using SACM alternate notation (as the first notation that they learned during experiment. The participants' average time when using SACM notation 09:31, and the participants' average time when using SACM alternate notation 11:54.

Timing - SACM notation Group B



Timing - SACM alternate notation Group B

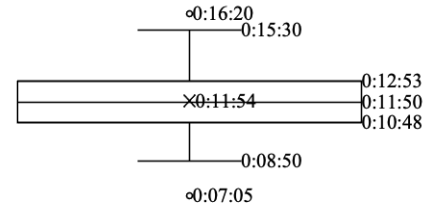


FIGURE 6.8: Performance results of Group B with respect to the times spent in creating the assurance case using SACM notation and SACM alternate notation (in hh:mm:ss)

Hypothesis testing

In this chapter, previously, we have formulated the hypothesis to evaluate the effectiveness of the SACM notation and SACM alternate notation in the sense of the time required by the participants to create an assurance case model.

Similar to the hypothesis testing to evaluate the accurate use of the notation, a Two-Sample t-Test for unequal variances was conducted to evaluate the effectiveness of the SACM notation and SACM alternate notation in the sense of the time required by the participants to create an assurance case model. We assessed the data distribution before conducting the test, and we found the data is not normally distributed as illustrated in Figure 6.9 using a histogram. However, according to the literature ([93]), it is not affecting the validity of the test.

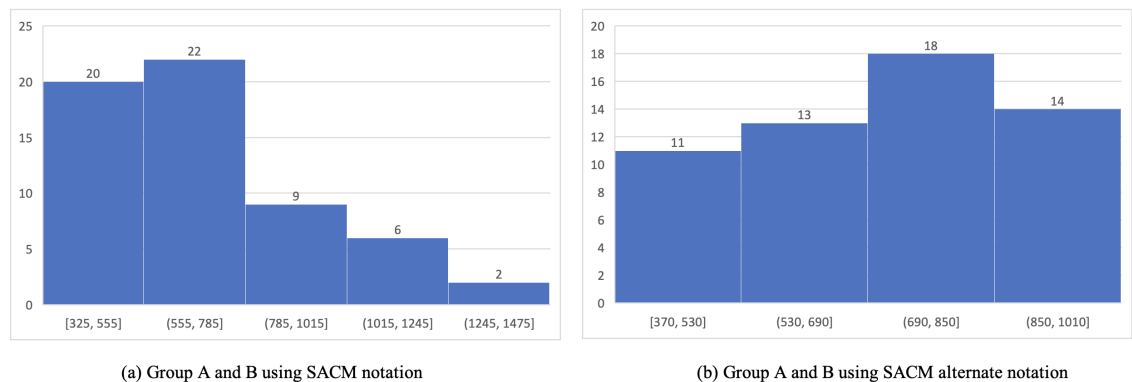


FIGURE 6.9: Data distribution Group A and B for the effectiveness of the notation study - time required to create an assurance case (using (a) SACM notation and (b) SACM alternate notation)

The result of the test is presented in Table 6.6.

TABLE 6.6: t-test results for the effectiveness the notation in terms of time required by the participants to create an assurance case using SACM notation and SACM alternate notation

Group	Notation	N	Mean (in seconds)	Variance	P value	Alpha
A+B	SACM	59	712.85	66375.82	0.46	0.05
	SACM alternate notation	56	709.00	30947.49		

The first column of Table 6.6 indicates the groups that participated in this study. The second column indicates the notations used by the participants to create the assurance case model. "N" in the third column indicates the number of participants involved in the study - 59 participants in total from Group A and B who involved in the assurance case model construction task using SACM notation, and 56 participants who used SACM alternate notation. The "Mean" column indicates the participants' average time to create an assurance case using a particular notation (presented in seconds). The "Variance" column indicates the variance from the gathered data of the participants' score in using each notation. The column "P-value" indicates the P-value associated with the mean generated from the participants' score using each notation. The "Alpha" column indicates the alpha number (margin of error), in this case, is 0.05.

As shown in Table 6.6, the average time spent by all the participants from both groups to create an assurance case model using SACM notation is 712.85 seconds and 709.00 seconds using the SACM alternate notation. In general, this result indicates that the participants slightly quicker in creating an assurance case using SACM alternate notation than using the SACM notation. The difference was only 3.85 seconds. This means the SACM alternate notation is slightly relatively easy to use than the SACM notation in terms of time required by the participants to create an assurance case model. In this case, the result indicates something that we are not expected. Therefore, we need to test whether the Null Hypothesis can be statistically accepted (Null Hypothesis stated that the time spent by the participants to create an assurance case using SACM alternate notation is better than SACM notation).

By conducting the t-Test, the resulting P-value is equal to 0.46, in this case, it is bigger than the Alpha number that we used in this test (0.05), and it is within the rejection region. Therefore, we may conclude that we cannot statistically accept that the SACM alternate notation is more effective than SACM notation in the context of time spent by the participants to create an assurance case model based on the given case study.

6.2.4.5 Identification of element variants

In this part, the result of study related to the following research question is discussed:

How effective is the produced SACM notation in the sense of its intuitiveness related to identifying the correct variants of the element by the participant?

The participants' responses regarding the questions related to identifying element variants of SACM notation and Alternate SACM notation were analysed to answer the above research question. All the participants from both groups were asked to identify element variants of SACM notation after learning and using the notation in the assurance case construction task. Similarly, they were also asked to identify the Alternate SACM notation element's variant after learning and using the notation. The

questions in this part were designed to help understand the intuitiveness of each notation element's visual representation.

All the Group A participants, 22 participants, were asked to identify the SACM element variants first and then continue to Alternate SACM notation. The participants in Group B, 32 participants, were asked to identify the Alternate SACM notation element's variants first and then the SACM notation element variants. The participants were allowed to choose more than one answers. They were also allowed to decide not to answer a particular question if they think there is no relevant answer provided in the given options.

Based on the participants' responses, we identified the effectiveness of the SACM notation in the sense of the visual representation intuitiveness in the context of the identification of element variants based on visual representation similarity is considered high for several elements; in average, more than 50% from a total of 54 participants were correctly identified the variants of the following elements:

- 'Claim'
- 'ArgumentPackage'
- 'defeated'
- 'abstract'
- 'asCited'
- 'needsSupport'

The participants were struggling to identify the variants of the following elements; on average, less than 50% from a total of 54 participants were correctly identified the variants of the following elements:

- 'assumed' element, especially, when the participants were provided by the information related to the 'assumed Claim', they experienced difficulty identifying the 'assumed Context' and 'assumed Evidence' as the variants of the 'assumed' element.
- 'axiomatic' element, in the context of when the participants were given the information related to the 'axiomatic Inference', they struggle to recognise the 'axiomatic Claim' as one of the 'axiomatic' element variants.

Relative to the Alternate SACM notation, the design of the SACM notation is less intuitive in the context of visual representation design similarity. This is indicated by the average result of the element variants identification of the Alternate SACM notation is greater than the result of the element variants identification of the SACM notation. For example, as shown in Table 6.7, the participants' responses

related to identifying ‘Claim’ element variants of the Alternate SACM notation, in average, is greater than the SACM notation. The comparison result of other elements can be read further in Appendix D.

TABLE 6.7: Comparison of participants’ responses related to the identification of ‘Claim’ element variants of SACM notation and Alternate SACM notation

Task	Identification of Claim variants			
Info given	Claim			
Element variants	Total participants (Group A + B): 54			
	SACM Notation		Alt. SACM notation	
	Count	%	Count	%
<i>AssumedClaim</i>	31	57%	34	63%
<i>AsCitedClaim</i>	29	54%	35	65%
<i>AxiomaticClaim</i>	36	67%	42	78%
<i>DefeatedClaim</i>	17	31%	33	61%
<i>NeedsSupportClaim</i>	31	57%	33	61%
<i>AbstractClaim</i>	28	52%	40	74%
Average		53%		67%

The result gathered from the element variants identification task indicates that the design decision made to create the SACM notation can still be improved in the context of enhancing the intuitiveness of the visual representation similarity. According to the visual notation design literature such as in [3], visual representation similarity can help notation users learn faster about the notation, especially to identify the related elements each other, e.g. semantically related. The result also indicates that the design might influence a notation’s effectiveness, in this case, both of the notations in this experiment designed using the same (proposed) visual notation design approach; however, created using different design decisions.

6.2.4.6 Utilisation of visual variables

Similar to the evaluation of the SACM notation relative to GSN, in evaluating the SACM notation relative to its Alternate notation, the intuitiveness of the visual variables used to create the notations was also being assessed. In this evaluation, the utilisation of visual variables Shape, Size, and Brightness were evaluated using several questions for the participants to answer.

The same questions as used in the evaluation of the SACM notation relative to GSN in assessing the intuitiveness of the utilisation of the visual variable were asked to the participants in this study:

- Four questions related to the utilisation of Shape (especially related to the use of arrow direction) and Position (for diagram layout)
- A question related to the utilisation of Size in the context of line thickness
- A question related to the role of Brightness

There were 54 participants in total from both Group A (22) and B (32). For each question, the participants were allowed to choose exactly one answer from the given options. They were allowed not to answer a particular question if they considered no suitable answer options were provided. The participants were also made aware that the diagram example presented in each question is not referring to a particular notation. This is due to reduce bias related to each question's participants' responses considering they participated in this task just after learning both SACM notation and Alternate SACM notation.

Based on the participants' responses related to the role of visual variable shape in representing relationship visual representation, in this context, a decorator such as an arrowhead can help participants to decide the conclusion Claim from two different Claims provided in the diagram. The position of the elements, vertically or horizontally relative to each other's position, might not affect participants' perception; the element that is pointed by the arrowhead was always selected as the conclusion element. This result supports our design decision in particular related to the SACM 'AssertedRelationship' type visual representation where a specific decoration such as an arrowhead used to indicate the target element of the relationship.

Related to the role of visual variable size in the context of line size thickness, based on the participants' responses, the line size thickness could provide emphasis to draw participants' attention, in this case, to indicate the conclusion element. Although in this context, the relationship that connects the element was not equipped with any decorator such as arrowhead to indicate the target or conclusion element. This result may help us to support our design decision particularly in the context of 'axiomatic' element where in this case a thick line is used as part of the 'axiomatic' element visual representation to indicate an emphasis that this element is declared to be axiomatically true and no further argumentation (elements) needed to support this element such as used to represent an 'axiomatic Claim'.

As for the effect of different type of brightness used in relationship visual representation, based on the participants' responses in this study, a light brightness when being used as part of the relationship visual representation (used in a specific line head type) might indicate the meaning of "Counter" relationship. In this context, the participants were previously informed that the dark brightness was used to indicate "Support"-type relationship. This result supports our design decision, particularly in the design of 'Counter'-type visual representation as used in 'Counter AssertedEvidence' relationship.

The result of the participants' responses regarding the utilisation of visual variables can be found in Appendix D.

6.2.4.7 Participants' perception towards the notations

In this part, the participants were asked to provide their opinion regarding their experience in learning and using the SACM notation and the Alternate SACM notation. There were in a total of 54 participants who involved in this part: 22 participants from Group A and 32 participants from Group B. They were asked to provide answer for the following questions:

1. Which notation do you considered as the notation that is easy to learn?
2. Which notation do you considered as the notation that is easy to use?
3. Which notation do you intend to use in practice when being asked to construct an assurance case?

Table 6.8 presents the summary of participants' responses from each group regarding their perceptions towards the experience in learning and using both SACM notation and Alternate SACM notation.

TABLE 6.8: Summary of participants' perception from each group: SACM notation and Alt. SACM notation

Group	Ease of learning	Ease of use	Intention to use
A (22 participants)	SACM (45%)	SACM (68%)	SACM (45%)
B (32 participants)	SACM (47%)	Alt. SACM (44%)	SACM (63%)

Based on participants' responses as presented in Table 6.8, most of Group A participants chose SACM notation as the notation that is both easy to learn (45%) and easy to use (68%). They also chose SACM notation as the notation that they intend to use in the future (45%). Group B participants chose SACM notation as the notation that is easy to learn (47%) and the Alternate SACM notation as the notation that is easy to use (44%). As for the notation that they intend to use in the future, most of them chose SACM notation (63%).

The result presented on Table 6.8 indicates if the participants were given a chance to learn and use both SACM notation and its alternate notation, most of the participants from groups have the intention to adopt the SACM notation in the future to create an assurance case model. However, some Group B participants believed that the Alternate SACM notation was relatively easy to use based on their experience during the experiment of the assurance case model construction task.

To summarise, as shown in Table 6.9, from all (54) participants in Group A and B who submitted their responses regarding their perception towards the notations that they have learnt and used, we conclude that most of the participants agreed that the SACM notation is relatively easy to learn and use than the Alternate SACM notation. They also intend to adopt the SACM notation as the notation that they are willing to use in the future when facing a similar problem (i.e. create an assurance case model).

TABLE 6.9: Summary of participants' perception: SACM notation and Alt. SACM notation

Group	Ease of learning	Ease of use	Intention to use
A+B (54 participants)	SACM (46%)	SACM (48%)	SACM (56%)

The result of the participants' responses regarding their perception towards the notations can be found in Appendix D.

6.3 Experienced user evaluation

In this section, the evaluation of the SACM notation based on the perspective of experienced assurance case users is discussed.

6.3.1 Study definition and planning

6.3.1.1 Objective

A survey involving experienced users was conducted to seek feedback from experienced assurance case stakeholder regarding the produced SACM notation. Another objective of the survey is to understand the experienced users' opinion regarding the usefulness of some new concepts (including the advantages and disadvantages of the notation) that are proposed by the SACM argumentation specification relative to their knowledge and experience related to the existing notation that they regularly used. The gathered feedback from the survey can be used as an input for the improvement of the SACM notation in the next notation development.

6.3.1.2 Survey design

A questionnaire was developed to conduct the survey which evaluates the produced SACM notation. The questionnaire was designed to allow participants to provide qualitative data. The qualitative data is useful to gain further insight from the participants; for example, why certain visual representations were preferred and why others were not preferred.

The questionnaire was divided into three sections. The first section contained questions related to the background of the participants, such as the domain of work, years of experience, and notation that regularly used.

The second section contained questions related to the clarity of the SACM notation relative to an existing notation. In this case, the GSN was used as the comparative notation. In this section, the participants were provided by two diagrams that illustrate the same assurance case: the first diagram was presented using GSN, and the second diagram was presented using SACM notation. The participants were instructed to observe both diagrams, and were asked to provide their feedback regarding the clarity of the notation.

The third section contained questions related to the usefulness of the SACM features, in this case, the elements that are not provided by other existing notations, such as ‘metaClaim’, ‘Counter Inference/Evidence’, and ‘ArgumentGroup’. The participants were also asked to provide comments regarding (the advantages and disadvantages of) the visual representation of these elements.

The survey was designed as an online survey where the information regarding the objective of the survey was informed to the invited participants via email along with the web address to the survey.

6.3.1.3 Participants selection criteria

The participants were selected through non-probabilistic-convenience sampling approach, which means the participants were selected based on their availability to participate in the survey. The survey was conducted from the point of view of experienced assurance case users with various background and experience in assurance case development and evaluation.

6.3.1.4 Pilot

Before conducting the main survey, we considered a pilot survey was necessary to capture the potential problems that might occur when conducting the main survey. Two participants were involved in the pilot survey. There were no problems identified based on the pilot, and the participants’ feedback was found to be relevant to the questions asked. Therefore, the main survey was conducted as planned.

6.3.2 Data analysis

6.3.2.1 Pre-processing

In total, we received 10 responses. Before analysis, we pre-processed the data to eliminate any unusable responses. We manually detected suspicious entries, in this case, we found 2 responses that only answered the demographics questions and did not answering any the other survey questions. Therefore, we excluded these 2 responses, and as the result, we used the remainder of the responses (8) for data analysis.

6.3.2.2 Analysis procedure

The data gathered from the survey was a qualitative data that consists of participants’ comments regarding the clarity of the notation, usefulness of the SACM features (represented in elements), and including the advantages and disadvantages of the created visual representations.

We applied the qualitative approach [94] to explore the participants’ feedback regarding the SACM notation. The participants’ responses were grouped into a specific category based on their similarity. For example, for the participants who mentioned ‘ArtifactReference’ as the element that they found to be troublesome, then this comment was grouped into the ‘ArtifactReference’ group comment. The

author conducted the grouping process with the help of other two PhD students, this is due to reduce bias in determining the category of the comments.

6.3.3 Survey Result

6.3.3.1 Demographics

Eight participants agreed to partake in the survey. As for the first demographic question, the participants were asked whether they have experience related to the assurance case notations. All of them claimed they have experience related to the assurance case notations. As a follow-up question, the participants were asked about the notation that they have experience with. All of the participants claimed they have experienced in using GSN, and 2 out of 8 participants claimed that they also have experienced in using CAE.

The participants were also asked about their years of experience related to the use of assurance case notation. The majority of the participants (6 out of 8) claimed they have 10 to 20 years of experience related to the assurance case notation. The remainder of the participants (2) claimed that they have more than 20 years experience. As the last demographic question, the participants were asked to provide the information related to their domain of work. As shown in Table 6.10, the participants who take part in this survey came from different domain of works. For example, 4 participants work in the aerospace area, 4 participants work in the defence area and 3 participants who work in the automotive area. The responses gathered from the demographic questions indicated that all the participants were relevant to be involved in the survey.

TABLE 6.10: Survey participants: demographics

Total Participant	8			
Assurance case notation experience	8	Years of experience?		Domain of work?
Which notation?		10 to 20	6	Automotive
	GSN	More than 20	2	Rail
	CAE			Aerospace
				Nuclear
				Medical
				Defence
				Maritime

6.3.3.2 Clarity of the notation

In this section, the participants were provided by two different assurance cases: A and B. Each assurance case was presented by two different diagrams: the first was presented using GSN, and the second using SACM notation. Figure 6.10 illustrates the assurance case A that presented using GSN involving elements such as Goal, Strategy, Context, and Solution.

In Figure 6.11 the same example from Figure 6.10 (assurance case A) is presented using the SACM Notation.

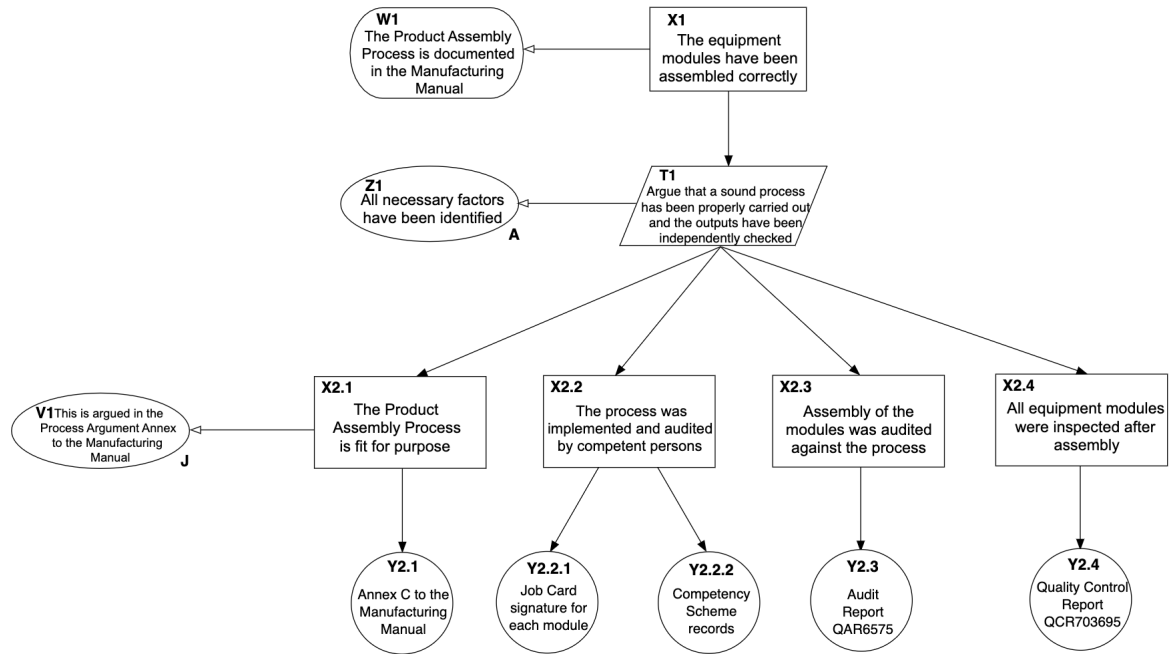


FIGURE 6.10: Assurance case A presented using GSN

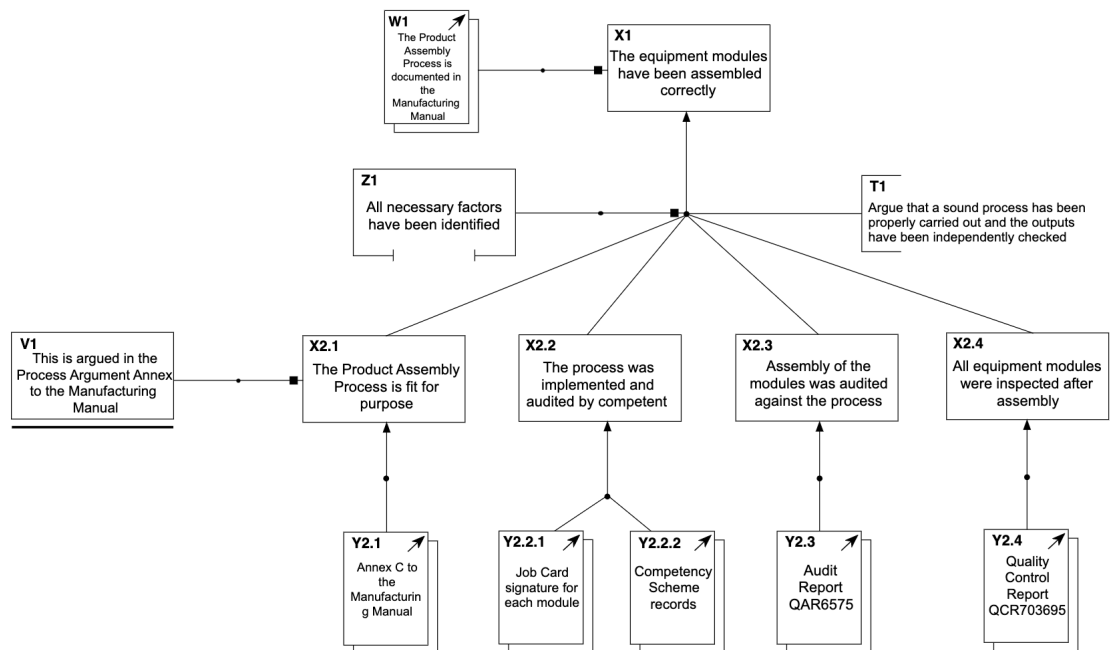


FIGURE 6.11: Assurance case A presented using SACM notation

Figure 6.12 illustrates the assurance case B that presented using GSN involving elements such as Away Goal (C2), Undeveloped Goal (C5), and Uninstantiated Goal (C4).

As for the assurance case B that is presented using SACM notation is shown in Figure 6.13 involving elements such as ‘asCited Claim’, ‘needsSupport Claim’, and ‘abstract Claim’.

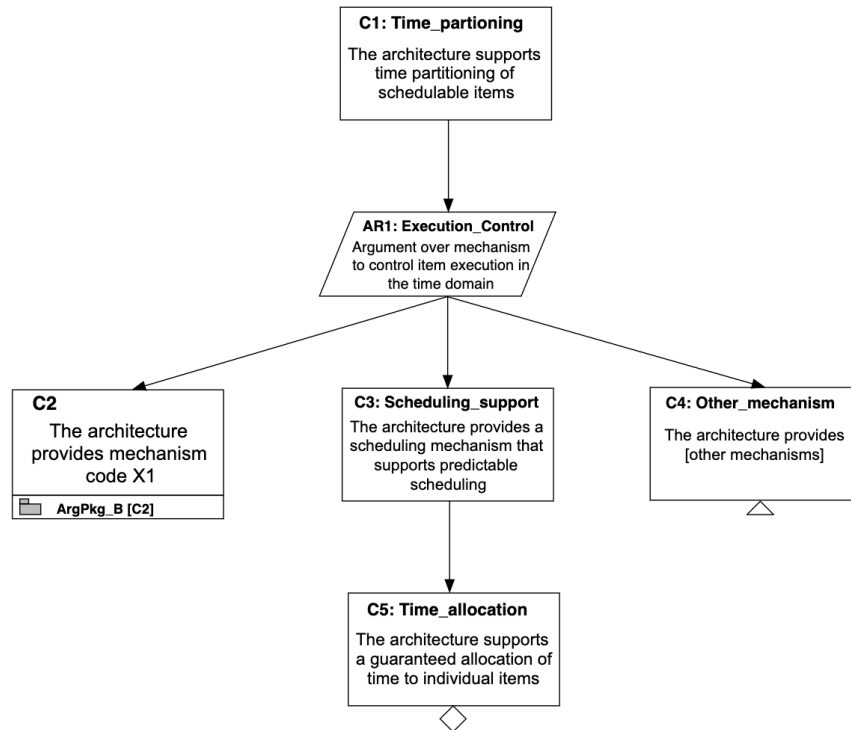


FIGURE 6.12: Assurance case B presented using GSN

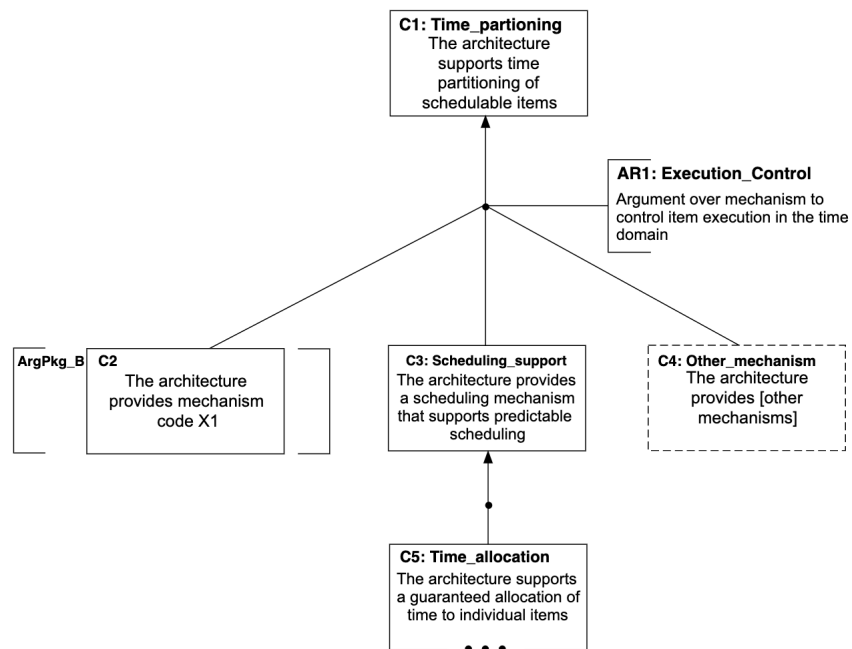


FIGURE 6.13: Assurance case A presented using SACM notation

After observing the given assurance cases, the participants were asked to provide their feedback by considering the following questions:

- *In what ways do you think SACM representation is better than GSN?*
- *In what ways do you think GSN representation is better than SACM?*

- *Do you find the SACM notation intuitive?*
- *Which elements of SACM do you find troublesome?*

Based on participants' responses after observing the assurance case A, at least two SACM notation elements have been mentioned to be better than GSN such as 'Axiomatic Claim' and 'Argument Reasoning'. After the participants observed the assurance case B, where in this case focuses about the use of different types of 'Claims' (or Goal in GSN), most of the participants stated that the majority of the SACM representation is better (or more intuitive) than GSN. This result indicated that the participants found the different types of 'Claim' in SACM notation are more intuitive than the GSN. For example, as stated by one of the participants:

"Actually in this case quick comparison between the two notations suggests the SACM representation is better than the GSN primarily because the notation is a more intuitive representation of the Away Goal (C2), Undeveloped Goal (C5), and Uninstantiated Goal (C4) in GSN."

The following is the other examples of participants' feedback regarding the representation of SACM notation that is better than GSN:

- *"It is even clearer in SACM than if you'd used the modular GSN symbol for away goal."*
- *"The universal use of rectangles is pragmatically a better approach than also using circles and ellipses, because it is more efficient use of space as text does not fit well into circles and ellipses."*
- *"The SACM is intuitive and perhaps focuses the author on the relationships between the Goal, Arguments/Justification to Evidence/Solutions by side-lining (for want of a better word) the Strategy."*
- *"The meaning of the annotations for uninstantiated and undeveloped goals are clear in SACM."*
- *"I think the SACM notation for Undeveloped and Unsubstantiated are stronger than the GSN counterparts in that they are markedly different and less likely to be misunderstood."*
- *"The use of the dotted lines in C4 is an improvement, as shows is not complete."*
- *"The SACM model is easier to understand and follow."*

We identified participant's feedback that stated He/She could not find any feature of SACM to be better than GSN, most likely due to a lack of understanding of SACM (notation). There was also a participant who stated that He/She could not find any significant differences between the assurance case presented in SACM notation and GSN.

After observing the assurance case A, we also identified at least three participants mentioned that GSN Context and Evidence relatively better than SACM 'ArtifactReference'. As follow is an example of the feedback:

"The difference between contextual information and evidence is not immediately obvious with this notation."

We assumed the participants struggle to differentiate between Context and Evidence in SACM notation due to their familiarity with GSN. In SACM, both Context and Evidence are defined in the metamodel as an Artifact. Therefore, as the notation designer, we need to develop an idea how to differentiate the use of an Artifact in the diagram so the notation user still can recognise when it is being used as evidence or context.

The participants also mentioned several elements such as Uninstantiated Goal, Strategy, and Away Goal of GSN are better represented than in the SACM notation after observing the assurance case B. The following is the example of the participant comment:

"I think the notation for Away Goal is probably easier to understand in GSN."

As follow, we summarised some of the participants feedback regarding the representation of GSN that is better than SACM notation and which element (or what aspect) of SACM representation that the participants found it troublesome:

- *"I am not as comfortable with all the embellishments between elements - both dots and boxes - I think the boxes are equivalent of context arrows in GSN, but I suspect some confusion when we are using the ACP notation in GSN."*
- *"I worry about what the dots are supposed to mean - they have meaning in pattern language GSN, but don't seem to have specific meaning in SACM, and they probably won't survive well in photocopies versions or re-scaled versions imported into document."*
- *"The use of the dot to join Y2.2.1 and Y2.2.2 may not be practical in real life."*
- *"The SACM elements are much more difficult to draw with non-specialist tools, such as Visio, powerpoint, lucid chart, etc."*
- *"I think the use of different shapes to represent different architectural elements to be clearer in GSN rather than the use of different rectangles in SACM."*
- *"Still struggling a bit with having strategy not as part of the logical flow and worry how readable it will be if there are lots of sub-claims/goals."*
- *"It is easier to follow the structuring of the GSN argument via strategy elements, in SACM, they appear "off to the side" as if there were a context element."*

Based on the participants' feedback, we considered further work is required to rethinking about the use of dot as part of the 'AssertedRelationship' visual representation since participants found the dots might cause trouble such as it might not survive well in photocopies versions or re-scaled versions imported into document. Some participants also mentioned about the use of 'ArgumentReasoning'

in SACM diagram is different in GSN. They claimed to be struggling to understand this application. This could be due to their familiarity with the GSN approach.

We also recognised the participant's feedback that mentioned the use of different shapes in GSN elements to be clearer than in SACM notation due to the use of different rectangles. The use of different rectangles in SACM results from the application of the proposed visual notation design. For example, all different types of 'Claim' were visualised using rectangle (with a specific type of decoration based on each 'Claim' semantic) due to consider the visual inheritance of the '(asserted) Claim'. We considered the visual representation similarity between related elements such as 'Claims' are important to show family resemblance among visual representations that have similar semantics.

Regarding the tools that can be used to create an assurance case using SACM notation, we recognise that it is likely to be hard to draw the visual representation of the SACM elements with non-specialist tools. A tool named ACME (Assurance Case Modelling Environment) has been developed by a tool developer that provides the SACM notation as one of their notation libraries that can be used to develop an assurance case diagram. This might be an alternative for the participants who interested to use the SACM notation for representing their assurance case.

To summarise, we received valuable feedback from the experienced assurance case users. Some participants considered that the SACM notation is more intuitive than GSN, especially related to the representation of different types of 'Claim' that are design to be similar. This indicates that the visual inheritance benefits the notation user specifically to show family resemblance among visual representations of semantically related elements. We also recognised and highly appreciates the participants' feedback related to the representation of SACM notation that still needs to be improved, such as the dots in 'AssertedRelationship'. The participants' feedback is the most important part that needs to be considered for further improvement of the SACM notation.

6.3.3.3 Usefulness of SACM features and their representation

In this section, the participants were asked to provide their feedback regarding the usefulness of the SACM features that are not provided by existing notations. This part of evaluation aims to trigger the participants to provide feedback regarding the visual representations of other SACM elements such as 'metaClaim', 'Counter Inference'/'Counter Evidence', and 'ArgumentGroup'. In this case, we introduced several SACM elements to the participants by providing the basic information (element's name, semantics, and visual representation) and an example of the application in a (assurance case) diagram. The participants were asked to assess the usefulness of the elements based on their experience; after that, they were asked to provide their comments regarding the element's visual representation's advantages and disadvantages. The elements that were evaluated in this section are: 'metaClaim', 'Counter Inference', 'Counter Evidence', 'Defeated Claim', 'ArgumentPackageInterface', and 'ArgumentGroup'.

Based on the responses that we received, most of the participants agreed that those elements are useful in the development of assurance cases. As presented in Table 6.11, we also identified participants who unsure regarding the usefulness of a particular notation. For example, some participants were not sure at which phase of development (of assurance case) that the ‘Counter Inference’ or ‘Counter Evidence’ would be most likely to be used. Another example, we also identified a participant that claimed He/She was never experiencing the needs for the ‘ArgumentPackageInterface’ in practice.

TABLE 6.11: The usefulness of SACM elements

Element	Response received	Usefulness		Comment Example
		Yes	Unsure	
MetaClaim	8	8	0	>Useful: "Yes, as its useful to be able relate other factors that have a bearing on the strength (or otherwise) of an assurance case."
CounterInference	7	6	1	>Useful: "Yes - as in reality there are 'pros' and 'cons' with all methodologies and it is sensible to provide opportunity to present such arguments to aid the overall case." >Unsure: "I'm not sure at which phase of development you would be most likely to record a criticism in such a way".
CounterEvidence	7	6	1	>Useful: "(It) can be useful when constructively reviewing the assurance case." >Unsure: "I'm not sure at which stage of the process you intend to record it this way."
DefeatedClaim	7	5	2	>Useful: "This is useful and done in practice e.g. traffic lights." >Unsure: "Not sure how often such a claim would be presented and used in reality."
Arg.Pkg.Interface	7	4	3	>Useful: "Yes, very useful once you understand the principles". >Unsure: "I'm unsure."
ArgumentGroup	7	4	3	>Useful: "Yes. Often need to group bits of an argument by owner, sub system, role etc." >Unsure: "Possibly, but only as a means of presentation to explain different facets of the argumentation strategy."

Regarding the visual representation of the elements, in general, some participants found the visual representation of the elements was intuitive and straightforward such as for ‘Counter Inference’, ‘Counter Evidence’, and ‘Defeated Claim. However, we also identified participants who provide input for the improvement of the elements’ visual representation. For example, some participants found the ‘metaClaim’ visual representation may not be self-explanatory and hard to distinguish with other relationship types. There were also participants who found the ‘ArgumentPackageInterface’ visual representation was not intuitive, and there were also participants who found the ‘ArgumentGroup’ visual representation could be hard to be used/draw in a diagram.

Table 6.12 presents some feedback from the participants regarding the advantages and disadvantages of the visual representation of SACM elements. The other participants’ feedback can be found in Appendix D.

6.3.4 Threats to validity

The threats to the validity of the study are summarised as follow:

- Internal validity

TABLE 6.12: Example of participants' feedback regarding the SACM visual representations

Element	Visual representation	
	Advantages	Disadvantages
MetaClaim	"Good VR (this is very simple and very effective. Something I could easily explain to the clients/stakeholders.)"	"The notation is subtle and may not be distinguishable from other link types if the scaling of presentation is not appropriate (e.g. if its too 'zoomed out')"
CounterInference	"The symbol and its use appears straight forward and clear."	"It's not very visually different to the other (relationship) elements."
CounterEvidence	"I think its good and like the way it is based on the counter inference notation being 'repurposed'."	"This is definitely going to confuse those with a GSN background. It's negative connotation is not intuitively clear from the symbol."
DefeatedClaim	"Good. Its obvious what it means."	"Could mistake it for being deleted rather than wrong/defeated."
Arg.Pkg.Interface	"Seems very intuitive to me, I have no problems with it."	"Symbols are not intuitive (the 'link' symbol is OK, but the interface is not. Perhaps its more familiar to those that use UML?)"
ArgumentGroup	"Even better if they were in colour."	"Looks scruffy, does not scale to more than two or three groups."

The survey was conducted to seek feedback from experience assurance case users. Their background related to the assurance case notations that might affect their responses are anticipated in this study. Some of the participants' responses were also stated that their feedback might be bias due to their familiarity to a certain notation. However, participants' feedback from this survey is highly appreciated as an input that can be used for further improvement of the SACM notation.

- External validity

The result gathered from the survey is in the context of the feedback from eight experienced assurance case users as the study participants. It is also unsafe to generalise the current result since a different and bigger sample of participants might provide different pattern and conclusion.

- Construct validity

To ensure that the survey questions mean to the participants what we presume they do, we conducted a pilot study. The pilot's result indicates that the participants understand the questions - the participant's responses were relevant to the given questions.

The survey is also equipped with the demographic questions in order to make sure that the participants were relevant to be involved in the survey - participant fit (ensuring that the participants are those from whom we wish to gain an understanding of assurance case notations).

- Conclusion validity

A general threat to conclusion validity is related to the number of samples of the survey. Only eight assurance case experienced users were involved in the survey. We admit that it is challenging to find participants from industries. However, we expected more participants to be involved

in this study that might provide more feedback and insight for improving the SACM notation.

6.4 Evaluation through tool support

The objective of this evaluation is to show that the created SACM notation can be implemented in a tool support. In this case, the SACM notation has been implemented in a tool called TouchDraw as one of its notation libraries. Touchdraw is a vector 2D drawing and diagramming application tool for iPad, Mac, Android (phones and tablets) ¹. The TouchDraw tool that implements the SACM notation has been used by the author to produce all the assurance case models related to this thesis. This indicates that the implementation of the SACM notation in the TouchDraw tool is useful for the development of assurance case models using SACM notation.

The screenshot shown in Figure 6.14 shows the use of the SACM notation within the TouchDraw tool.

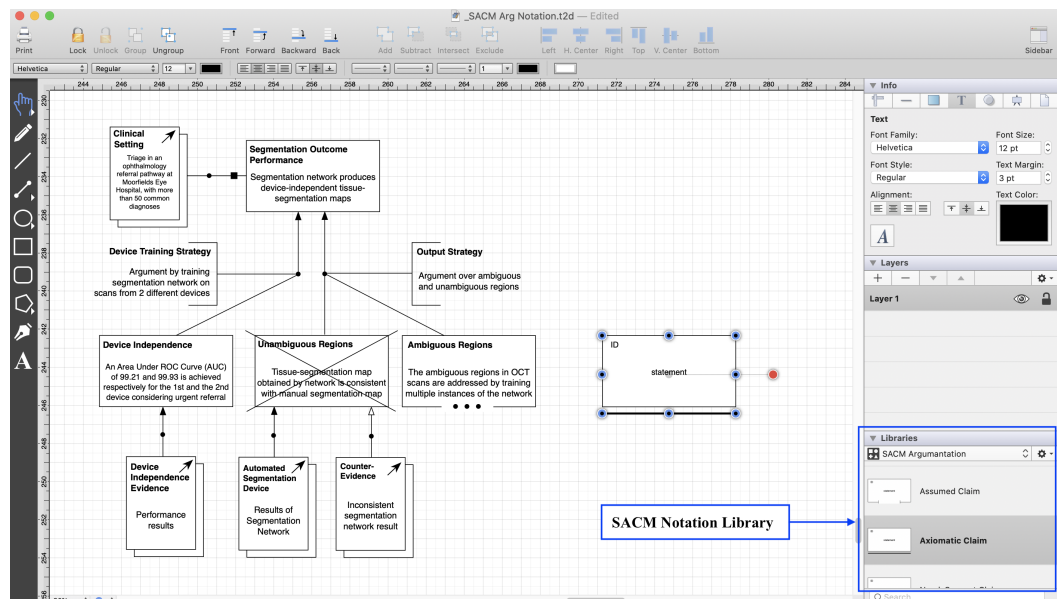


FIGURE 6.14: Implementation of the SACM notation in TouchDraw tool as one of its notation library

Other than being implemented as the notation library in TouchDraw, the SACM notation is also being implemented as part of the Assurance Case Modelling Environment (ACME) tool. ACME was developed by an independent tool developer with the aim to support the development of the assurance case model using notation such as SACM and GSN. The users of ACME are also able to transform their GSN models to SACM based on the model-to-model transformation [14].

¹<https://www.elevenworks.com/touchdraw>

6.5 Evaluation by international standard body

The created SACM notation has been submitted and reviewed by the Object Management Group (OMG) committee to be published as part of the updated SACM 2.1. version. As a result of the review process, the proposed SACM notation has been approved and published (on April 2020) as part of the updated SACM 2.1 specification. The published version of the SACM notation can be accessed in the following link: <https://www.omg.org/spec/SACM/About-SACM/>.

Figure 6.15 presents the screenshot from the OMG website regarding the publication of the SACM 2.1.

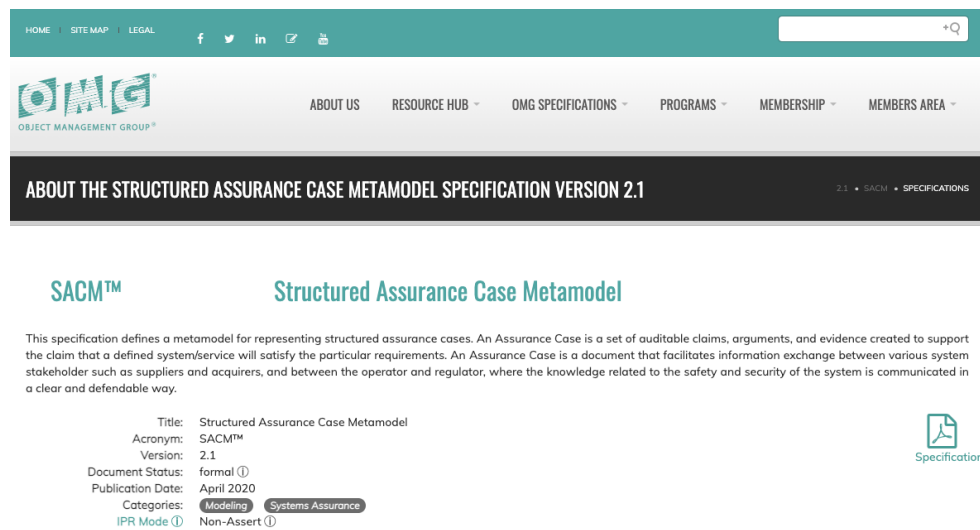


FIGURE 6.15: SACM 2.1 equipped by the proposed notation

6.6 Summary and conclusion

In this chapter, we have discussed the evaluation of the SACM notation developed based on the proposed visual notation design approach. The evaluation consists of several parts with a different objective:

1. Evaluation of the effect of the design decisions made during the development of SACM notation. This evaluation aims to investigate the effectiveness of the design decisions made during the development of the SACM notation relative to an alternate notation developed using the same approach, however, with different design decisions with SACM notation.
2. Evaluation involving experienced assurance case users with the aim to seek feedback based on their experience using existing notations.
3. Evaluation through tool support with the aim to show that the created SACM notation can be implemented in tool support to develop assurance case diagrams.

4. Evaluation of the SACM notation by the standard body (OMG Committee). This evaluation is part of the process to publish the resulting SACM notation as part of the latest version of SACM 2.1. specification.

Related to the evaluation of the effect of the design decisions made during the development of SACM notation, an empirical study has been conducted to evaluate the SACM notation to investigate whether the design decision made during the development of SACM notation might affect its (notation) effectiveness. The study involved 60 university students as novice participants. They were asked to participate in several evaluation tasks. In general, the result of the evaluation shows that the design decision may affect the effectiveness of a notation, in this case, based on the assurance case construction task, the effectiveness of the SACM alternate notation is better than the SACM notation in terms of the accurate use of the notation by the participants in creating assurance case model. The timing data result also shows that the participants relatively quicker to develop an assurance case using SACM alternate notation than SACM notation, although the time difference is only 0.04 second. Moreover, none of the participants who get a perfect score (100) in creating an assurance case using SACM alternate notation while 9 participants get a perfect score in creating an assurance case using SACM notation.

Based on the participants' attitude regarding the notations they have learned and used during the evaluation session, most participants believed that the SACM notation is relatively easy to learn and used than SACM alternate notation. They also have the intention to use the SACM notation than SACM alternate notation in the future when facing similar problems (i.e. creating assurance cases). Based on the evaluation result, we may also conclude, although the result of the participants' performance when using the SACM alternate notation is better than using the SACM notation, the participants tend to choose SACM notation as the notation that is easy to learn and use. Moreover, they have the intention to adopt SACM notation than the alternate notation.

Based on this evaluation, we also identified several elements of SACM notation required further work for further improvements. For example, the 'ArgumentPackage' visual representation's intuitiveness due to the low evaluation result received in visual representation intuitiveness task. The intuitiveness design similarity between visual representations is also needed further works such as for 'assumed' and 'axiomatic' elements.

To summarise, the result of the evaluation related to the effect of the design decision made during the development of SACM notation involving novice users shows that the design decision made for the SACM notation is not acceptable. This is indicated by the empirical study result that shows, in general, the effectiveness of the SACM notation is less than the effectiveness of the SACM alternate notation. In other words, the design decision made for the SACM alternate notation is better than the SACM notation. Based on this result, we may conclude that:

- The design decision made for the development of SACM notation still can be improved.

- The design decision made by the notation designer as part of the proposed visual notation approach can influence the effectiveness of the created notation.

Regarding the evaluation involving experienced assurance case users, an empirical study was also conducted to seek experienced assurance case users feedback regarding the produced SACM notation. There were eight participants involved in this evaluation. Most of them have experienced in assurance case domain between 10 to 20 years, and all of them have prior experience using GSN. The participants were asked about the clarity of the SACM notation relative to existing notation (GSN), and the intuitiveness (including the usefulness) of several SACM elements that are currently not provided by other existing notations, e.g. ‘metaClaim’, Counter-evidence/argument.

The evaluation results show that some participants provided positive feedback regarding the clarity of SACM notation, such as for the different types of ‘Claim’ in SACM is represented more intuitive than in GSN. We also received valuable feedback regarding the SACM elements that need further improvements, especially in the context of elements of GSN that are better represented than in SACM notation, e.g. Away Goal in GSN is easier to understand than in SACM notation.

We also received valuable feedback from the experience assurance case users regarding the intuitiveness of other SACM elements’ visual representation, such as ‘metaClaim’, ‘CounterInference’, and ‘CounterEvidence’. Some of the feedback received, for examples, the visual representation of the ‘metaClaim’ is very simple and effective, the ‘CounterInference’ the visual representation is straight forward, ‘ArgumentGroup’ looks scruffy and does not scale to more than two or three groups. Most of the participants also stated that SACM elements such as ‘metaClaim’, ‘CounterInference’, and ‘CounterEvidence’ are useful in the development of assurance cases.

To summarise, relative to the evaluation involving novice users, the experienced assurance case users are generally more favourable of SACM than the novices. We considered the feedback from the experienced users as valuable input for further improvement of SACM notation.

Regarding the evaluation of SACM notation through tool support implementation, the developed SACM notation has been implemented in a tool called Assurance Case Modelling Environment (ACME) to support the development of the assurance case model notation such as SACM and GSN. The SACM notation has also been implemented as one of the notation libraries in a TouchDraw tool. All the assurance case models developed in this thesis were created using the SACM notation implemented in TouchDraw. This indicates that the implementation of the SACM notation in a tool is useful for the development of assurance case models using SACM notation.

Apart from the above evaluations, the developed SACM notation has also been evaluated by an international standard body, in this case, is the Object Management Group, to be included in the updated version of SACM specification. As a result of the evaluation, the developed SACM notation is approved and included in the published version of the SACM 2.1 specification.

Chapter 7

Conclusions

7.1 Thesis contributions

An approach to developing an assurance case notation based on a defined metamodel has been defined in this thesis. The approach provides systematic guidance to develop a notation for assurance case notation that also has the potential to be applied to develop a notation in other domains. In this thesis, the proposed approach was used to develop the SACM argumentation notation. The evaluation of the proposed approach, including the produced SACM notation, was discussed in this thesis. The work presented in the thesis provides answers to research questions defined in Chapter 1. To summarise, the contributions of this thesis are focused in the following areas:

7.1.1 An approach to develop a visual notation based on a defined metamodel

Due to the lack of a systematic approach that can be used to develop a notation based on a defined metamodel, one of the aims of this thesis is to develop an approach that can be used to create a notation based on a metamodel (answer for **RQ2**). The proposed approach consists of four steps developed by considering the metamodel structure as the primary input and existing notation design literature such as visual variables, the Physics of Notation design principles, and notation usability factors.

The first step of the approach is *Create a design path* where the input for this step is a metamodel, and the output is a Design path that defines the design order of the metamodel elements. The second step is *Identify design constraints*. It focuses on the identification of design constraints that will be adopted in the development of the notation. Some examples of the design constraints include visual variables selection, considering existing notations and notation design principles, and notation usability factors. The second step's output is the list of the design constraints that are considered (by the notation designer) to be used in the creation of the visual notation.

The next step focuses on the creation of the visual representation of the identified metamodel elements. This step's input is the design path (output Step 1) and the identified design constraints (output Step 2). The development of the visual representation of an element considering the order defined in the design path. It is also considered the identified design constraints such as using a particular visual variable for a particular element. The development of a visual representation is also can be influenced by another visual representation(s). This is due to the application of *visual inheritance* process that is defined as part of the proposed approach. Visual inheritance makes it possible for two or more elements to have a similar visual representation. We considered visual representation similarity is important to show family resemblance among (related) elements. The output of the third step is the created visual representations.

As the last step, it is important to document all the design made during each visual representation creation. We learned from the literature that lack of visual representation design documentation might provide difficulty for the notation designer in improving the notation in the future [3, 17]. The last step in the proposed approach, specifically emphasises the development of the visual representation design documentation. In this context, a template for documentation is provided that can be used for the notation designer called as *visual representation identity template*.

By the development of this approach, hopefully, it could contribute to the continues work in the visual notation domain study.

7.1.2 A visual argumentation notation for the Structured Assurance Case Metamodel

The development of a visual assurance case notation of SACM is one of this thesis's contributions (answer for **RQ1**). The produced SACM argumentation notation is developed based on the proposed visual notation design approach defined in this thesis.

In the development process of SACM notation, input from notation stakeholders have been considered, such as developing a particular visual representation that needs to consider existing notation, such as UML. The structure of the SACM argumentation metamodel was also considered in the development of SACM notation due to considering the order of creating an element's visual representation. The SACM notation is created by considering design similarity among related elements (that has similar semantics) in order to increase the intuitiveness of the visual representations. Several design constraints have also been considered in the development of SACM notation, such as adopting visual variables as the visual notation building blocks, considering the similarity with existing notations, adopting existing notation design principles, and considering the notation usability factors.

The created SACM argumentation notation has been presented in Chapter 4, including its development process.

7.1.3 Empirical evaluation of the SACM argumentation notation

As part of the work presented in this thesis, several evaluation studies have been conducted to assess the effectiveness of the produced SACM notation (answer for **RQ3**). One of the evaluations is related to the empirical study involving the university students to assess the SACM notation's effectiveness relative to an existing assurance case notation (GSN). The result of this evaluation suggests that an improvement is necessary to improve some of the SACM element's visual representations, such as the intuitiveness of 'ArtifactReference' and 'ArgumentPackage' visual representation, and also the easy to draw aspect of the 'AssertedRelationship' types and the 'ArtifactReference' element.

Another evaluation has also been conducted to assess the effect of a design decision made in SACM notation development. This evaluation also involving university students with an aim to seek objective feedback. An alternate SACM notation was created by following the proposed visual notation design approach, however, with different design decisions with the proposed SACM notation. This evaluation shows that the alternate SACM notation produced several visual representations that are more effective than the proposed SACM notation. This indicates that the design decision might influence the effectiveness of a notation.

Other than evaluating the produced SACM notation from the point of view of University students as novice participants, we considered necessary to gain feedback from experienced assurance case users to share their perspective regarding the produced SACM notation. An empirical study has been conducted involving eight experienced assurance case users from different work background. The study was conducted as an online survey seeking participants' feedback regarding the intuitiveness and the usefulness of some feature of SACM notation that is not facilitated by other existing notations. The survey results show that most of the participants found some SACM element's visual representations are intuitive, such as different types of 'Claim' due to the application of visual inheritance approach. The participants also suggest that improvement is considered necessary for several elements such as the 'AssertedRelationship' types and the 'ArtifactReference' element. The result also indicates that most of the participants' agreed that the SACM notation features are useful for the development of assurance cases.

The empirical studies conducted as part of the work presented in this thesis are considered one of the contributions of this thesis. The evaluation result is expected to be considered as an input for further development of SACM notation.

7.2 Challenges and limitations

In this section, we highlight the challenges and limitations related to the thesis. We outlined the following items based on our experience in developing the proposed visual notation design approach and conducting the evaluation regarding the application of the proposed approach:

- As presented in Chapter 4 in this thesis, the proposed visual notation design approach has been used only for the development of the SACM notation. We considered further work is necessary to explore the proposed approach's applicability in different cases, e.g. different metamodel and/or different domains.
- In SACM notation development, the design decision made considering multiple inputs from different stakeholders such as expert users and tool developer. The produced SACM notation, as presented in Chapter 4, is the result of our first iteration in the development of the notation. The design decision made might be not acceptable; therefore, the empirical studies' feedback can be considered an input to refine the notation in the next iteration of SACM notation development.
- In the assurance case construction task as part of the empirical study involving university students, only the basic elements of the SACM notation have been used and evaluated. This is because of considering the duration of the experiment and the learning load of the participants. Therefore, further work is considered necessary to evaluate other SACM notation elements' effectiveness in a similar task (i.e. assurance case model construction task).
- There is a limitation regarding the evaluation of the proposed visual notation design approach. In this case, we were planned to invite individuals as notation designers to use the proposed approach to develop a notation for a particular metamodel. This evaluation aimed to seek feedback from a notation designer's perspective after adopting the proposed notation design approach to create a visual notation for a defined metamodel. However, we found it difficult to find an individual who is willing to participate in this evaluation type. Therefore, this evaluation could not be executed and could not be presented as part of this thesis. We aim to conduct this type of evaluation as part of the future work related to this thesis.

7.3 Future work

We considered the following areas presented in this section as subjects of future work related to this thesis.

7.3.1 Generality of the proposed visual notation design approach

One of this thesis's contributions is developing an approach to creating a visual assurance case notation based on a defined metamodel. The application of the proposed approach has been demonstrated in this thesis, specifically in the development of SACM argumentation notation. There is potential for the proposed visual notation design approach to be applied to develop other notations. In this case, it is possible to explore the proposed approach's applicability, such as developing a notation for another assurance case metamodel or developing a notation for another metamodel in other domains.

7.3.2 Refinement of the SACM notation

As part of the work presented in this thesis, an empirical study has been conducted to assess the effectiveness of the produced SACM notation. The evaluation is done through a comparison with SACM alternate notation created by following the proposed visual notation design approach, however, with a different design decision with the proposed SACM notation. The evaluation result shows that there is room for improvement to increase SACM notation's effectiveness, for example, by taking different design decisions in the development of future SACM notation (e.g., changing the decision regarding the use of a dot as part of the visual representation of Asserted Relationship elements).

A survey involving experienced assurance case users has also been conducted to seek feedback from experienced users. The result could be used as a valuable input to improve the SACM notation.

7.3.3 Experiment involving a larger number of experienced assurance case users with more complex examples

As part of the evaluation of this thesis, we have conducted empirical studies to assess the effectiveness of the resulting SACM notation. In one of the studies that have been conducted, we invited experienced assurance case users to provide feedback regarding the proposed SACM notation by showing them some examples of the application of the notation. However, there were only eight experienced users who participated in our study.

We considered it could be helpful to provide more insight regarding the effectiveness of the proposed notation by inviting a larger number of experienced assurance case users from multiple domains to use the proposed SACM notation with more complex examples. By doing this kind of experiment, we also could collect feedback from them regarding the usefulness of the SACM notation features (e.g., complexity management and dialectical argumentation) in practice.

7.4 Overall conclusions

In this thesis, we have developed an approach that can be used to create a visual notation based on a defined metamodel. This approach is developed to address the gap in the visual notation development research domain where currently there is a lack of approaches that the notation designers can adopt to develop a visual notation based on a defined metamodel.

The application of the proposed approach has been discussed and presented in this thesis to develop the SACM notation. When applying the approach, many design decisions are required to develop a notation. We discovered that the notation development process in this context is sensitive to the design decisions made by the notation designer when applying the proposed approach. This is indicated by the result of the empirical study that shows the design decisions made for the SACM alternate notation is slightly better than the design decisions made for the SACM notation. Both of the notations were

developed using the same (proposed) approach; however, they are different in terms of the design decisions made. The design decisions made for the SACM alternate notation in this context still within the scope of the proposed visual notation design approach where the design decisions were made based on the existing notation design theories and consider the notation usability and the adoption of existing notations in the domain.

Based on the empirical study, we also discovered that the experience assurance case users are generally more favourable of SACM notation than the existing notation in terms of the intuitiveness aspect of the notation. The experienced assurance case users also found that the SACM notation is useful to develop assurance cases. Furthermore, we also received valuable feedback from the empirical study that can be used for further development of the SACM notation.

To summarise, we consider that the proposed approach has the potential to be adopted to develop any visual notation based on a defined metamodel. In this thesis, the proposed approach has been applied to develop the SACM notation. The produced SACM notation is found to be intuitive and useful by experienced assurance case users. Furthermore, the SACM notation can be adopted by the practitioners to develop assurance cases because of the produced SACM notation, that is developed using the proposed visual notation design approach, has been adopted by the international standard body - Object Management Group (OMG) - where the SACM notation has been included and published as part of the updated version of the SACM 2.1. standard.

Appendix A

Visual representation identity of SACM notation

A.1 ‘Claim’ and its variants

A.1.1 ‘Claim’

TABLE A.1: ‘Claim’ visual representation identity

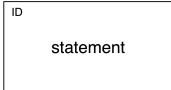
Visual representation identify of		‘Claim’
1	Visual representation	
2	Semantics	‘Claims’ are used to record the propositions of any structured argument contained in an ‘ArgumentPackage’.
3	Visual variable used	‘Claim’: <ul style="list-style-type: none"> • Shape: ‘Rectangle’ • Brightness: ‘Light’ (background of the Rectangle) • Location: ‘Vertical/horizontal’ relative to its connected element. • Texture: ‘Solid line’ used as the line texture type of the Rectangle. • Size: "1pt" as the size of the line that form the rectangle. Inherited from the ‘gid’ attribute design constraint: <ul style="list-style-type: none"> • Location: ‘Top-Left’ within the element.
4	Visual inheritance info	Inherited design constraint of ‘gid’ attribute.
5	Design rationale	
	Avoiding clash with the existing notations	The visual representation of the ‘Claim’ is adopted from the visual representation of the GSN Goal. This is due to provide a clue to the notation user who are familiar with this visual representation (reduce learning effect).
	Adopting existing visual notation design principles	
	Semiotic clarity	The Rectangle is only used for the visual representation of the ‘Claim’.
	Semantic transparency	Considered as an index visual representation in order to provide a clue to the users who familiar with the existing notation (GSN Goal).
	Perceptual discriminability	The visual representation of the ‘Claim’ can be differentiated with other visual representations.

Table A.1 continued from previous page

Visual representation identify of		'Claim'
	Redundant coding	The visual representation of the 'Claim' is created using multiple visual variables.
	Notation Usability	
	Easy to draw	The visual representation of the 'Claim' is considered relatively easy to draw.
	Use of colour	Not using Colour to create the visual representation of the 'Claim'.
	Tool support implementation	The visual representation of the 'Claim' is considered feasible to be implemented in a tool support.
	Other constraints (if any)	N/A.
6	Compositional rules when used in a diagram	<p>The main 'Claim' is located at the top-level of the diagram. It can be supported by other 'Claims' which must be placed below the main 'Claim' and must be connected via 'AssertedInference' relationship.</p> <p>A 'Claim' can be supported by another 'Claim', in this context, the supporting 'Claim' is used as a 'Claim' that provides context to the supported 'Claim'. The supporting 'Claim' must be located on the right/left side of the supported 'Claim' and must be associated via 'AssertedContext' relationship.</p> <p>A 'Claim' can be supported by more than one 'ArtifactReferences' that are used as a reference to an evidential information. In this case, the 'ArtifactReferences' must be located below the 'Claim' and must be associated via 'AssertedEvidence' relationship.</p> <p>In another case, more than one 'ArtifactReferences' can also be used as a reference to an contextual information to scope and provides context to a particular 'Claim'. In this case, the 'ArtifactReferences' must be located horizontally relative to the 'Claim' and must be connected via 'Asserted-Context' relationship.</p> <p>A 'Claim' can also be used as a supporting element for an 'AssertedRelationship' type (e.g. 'AssertedInference'). In this case, an 'AssertedInference' relationship must be used to connect the 'Claim' and the supported 'AssertedRelationship' type.</p>

A.1.2 'needsSupport Claim'

TABLE A.2: 'needsSupport Claim' visual representation identity


Visual representation identify of		'needsSupport Claim'
1	Visual representation	
2	Semantics	A 'Claim' that is intentionally declared as requiring further evidence or argumentation
3	Visual variable used	<p>Inherited from the design constraints of a 'Claim':</p> <ul style="list-style-type: none"> • Shape: 'Rectangle' • Brightness: 'Light' - background of the Rectangle. • Location: 'Vertical' or 'horizontal' relative to its connected element; 'Top-Left' within the rectangle for the element ID (inherited from 'gid' attribute). • Texture: 'Solid line' - line texture of the Rectangle. • Size: "1pt" - the size of the line that form the rectangle.

Table A.2 continued from previous page

Visual representation identify of		'needsSupport Claim'
		Inherited from the 'needsSupport' design constraints: <ul style="list-style-type: none"> • Shape: '3 Circles' • Brightness: 'Dark' • Location: 'Bottom part of the rectangle'. • Size: '1pt' as the size of the line that form the Circle; '0.25' cm for the width and height of the circles (adjustable relative to the application in a diagram).
4	Visual inheritance info	Inherited design constraints of 'Claim', 'gid', and 'needsSupport'.
5	Design rationale Avoiding clash with the existing notations Adopting existing visual notation design principles Semiotic clarity Semantic transparency Perceptual discriminability Redundant coding Notation Usability Easy to draw Use of colour Tool support implementation Other constraints (if any)	No existing notation in the same domain is used the visual representation of the 'needsSupport Claim'. The three dots placed on the bottom part of the rectangle is used only for representing the 'needsSupport Claim'. Considered as an index visual representation for the user who already learn about the SACM 'Claim' and 'needsSupport'. The visual representation of the 'needsSupport Claim' can be differentiated with other created visual representation such as 'Claim'. The visual distance between them is not equal to zero. The visual representation of the 'needsSupport Claim' is created using multiple visual variables. The visual representation of the 'needsSupport Claim' is considered relatively easy to draw. Not using Colour to create the visual representation of the 'needsSupport Claim'. The visual representation of the 'needsSupport Claim' is considered feasible to be implemented in a tool support. N/A.
6	Compositional rules when used in a diagram	A 'needsSupport Claim' can be used as a supporting 'Claim' that supported another 'Claim' or a particular type of 'AssertedRelationship type'. It can be placed below the supporting element if the 'needsSupport Claim' supporting another 'Claim', 'AssertedContext', or 'AssertedArtifactContext' relationship. In such cases, the relationship between the 'needsSupport Claim' and its targeted element must be declared using an 'AssertedInference' relationship. When a 'needsSupport Claim' is used as a 'Claim' that provides context to another 'Claim', 'AssertedInference', or 'AssertedEvidence' relationship, it must be located horizontally (left/right side) relative to its targeted elements. The relationship between the 'needsSupport Claim' and its targeted element must be declared using an 'AssertedContext' relationship. A 'needsSupport Claim' must not be supported by other elements in terms of elements that need to be attached below the 'needsSupport Claim' due to consider its semantics as a 'Claim' that requiring further support from other elements.

A.1.3 'assumed Claim'

TABLE A.3: 'assumed Claim' visual representation identity

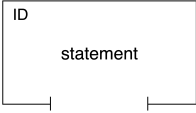
Visual representation identify of		'assumed Claim'
1	Visual representation	
2	Semantics	A 'Claim' that is intentionally declared without any supporting evidence or argumentation.
3	Visual variable used	Inherited from the design constraints of a 'Claim': <ul style="list-style-type: none"> • Shape: 'Rectangle' • Brightness: 'Light' - background of the Rectangle. • Location: 'Vertical/horizontal' relative to its connected element; 'Top-Left' within the rectangle for the element ID (inherited from 'gid' attribute). • Texture: 'Solid line' - line texture of the Rectangle. • Size: "1pt" - the size of the line that form the rectangle.
		Inherited from the 'assumed' design constraints: <ul style="list-style-type: none"> • Shape: 'A line with a gap in the middle of the line (with two vertical lines at the line-end where the line gap is)' • Location: 'Bottom part of the rectangle' when being applied in a 'Claim'. • Size: '1pt' as the size of the line. • Texture: 'Solid' for the texture of the line.
4	Visual inheritance info	Inherited design constraints of 'Claim', 'gid', and 'assumed'.
5	Design rationale	
	Avoiding clash with the existing notations	No existing notation in the same domain is used the visual representation of the 'assumed Claim'.
	Adopting existing visual notation design principles	
	Semiotic clarity	A rectangle with a line gap placed on the bottom part of it is only used for representing the 'assumed Claim'.
	Semantic transparency	Considered as an index visual representation for the user who already learn about the SACM 'Claim' and 'assumed'.
	Perceptual discriminability	The visual representation of the 'assumed Claim' can be differentiated with other created visual representation such as 'Claim' and 'needsSupport Claim'.
	Redundant coding	The visual representation of the 'assumed Claim' is created using multiple visual variables.
	Notation Usability	
	Easy to draw	The visual representation of the 'assumed Claim' is considered relatively easy to draw.
	Use of colour	Not using Colour to create the visual representation of the 'assumed Claim'.
Tool support implementation	The visual representation of the 'assumed Claim' is considered feasible to be implemented in a tool support.	
Other constraints (if any)		N/A.

Table A.3 continued from previous page

Visual representation identify of		‘assumed Claim’
6	Compositional rules when used in a diagram	<p>An ‘assumed Claim’ can be used as a supporting ‘Claim’ that supported another ‘Claim’ or a particular type of ‘AssertedRelationship type’. It can be placed below the supporting element if the ‘assumed Claim’ supporting another ‘Claim’, ‘AssertedContext’, or ‘AssertedArtifactContext’ relationship. In such cases, the relationship between the ‘assumed Claim’ and its targeted element must be declared using an ‘AssertedInference’ relationship.</p> <p>When an ‘assumed Claim’ is used as a ‘Claim’ that provides context to another ‘Claim’, ‘AssertedInference’, or ‘AssertedEvidence’ relationship, it must be located horizontally (left/right side) relative to its targeted elements. The relationship between the ‘assumed Claim’ and its targeted element must be declared using an ‘AssertedContext’ relationship.</p> <p>An ‘assumed Claim’ must not be supported by other elements in terms of elements that need to be attached below the ‘assumed Claim’ due to consider its semantics as a ‘Claim’ that is declared without any supporting evidence/argumentation.</p>

A.1.4 ‘axiomatic Claim’

TABLE A.4: ‘axiomatic Claim’ visual representation identity


Visual representation identify of		‘axiomatic Claim’
1	Visual representation	
2	Semantics	A ‘Claim’ that is being declared as axiomatically true, so that no further argumentation is needed.
3	Visual variable used	<p>Inherited from the design constraints of a ‘Claim’:</p> <ul style="list-style-type: none"> • Shape: ‘Rectangle’ • Brightness: ‘Light’ - background of the Rectangle. • Location: ‘Vertical/horizontal’ relative to its connected element; ‘Top-Left’ within the rectangle for the element ID (inherited from ‘gid’ attribute). • Texture: ‘Solid line’ - line texture of the Rectangle. • Size: “1pt” - the size of the line that form the rectangle. <p>Inherited from the ‘axiomatic’ design constraints:</p> <ul style="list-style-type: none"> • Shape: ‘A line’. • Location/Position: ‘Below’ the rectangle. • Size: ‘3pt’ as the size of the line; for the length of the line is relatively following the size of the rectangle that is placed above the line. • Texture: ‘Solid’ for the texture of the line.
4	Visual inheritance info	Inherited design constraints of ‘Claim’, ‘gid’, and ‘axiomatic’.
5	Design rationale	
	Avoiding clash with the existing notations	No existing notation in the same domain is used the visual representation of the ‘axiomatic Claim’.
	Adopting existing visual notation design principles	
Semiotic clarity	The ‘axiomatic Claim’ visual representation is only used for representing the ‘axiomatic Claim’.	

Table A.4 continued from previous page

Visual representation identify of		'axiomatic Claim'
	Semantic transparency	Considered as an index visual representation for the user who already learn about the SACM 'Claim' and 'axiomatic'.
	Perceptual discriminability	The visual representation of the 'axiomatic Claim' can be differentiated with other created visual representation such as 'assumed Claim' and 'needsSupport Claim'.
	Redundant coding	The visual representation of the 'axiomatic Claim' is created using multiple visual variables.
	Notation Usability	
	Easy to draw	The visual representation of the 'axiomatic Claim' is considered relatively easy to draw.
	Use of colour	Not using Colour to create the visual representation of the 'axiomatic Claim'.
	Tool support implementation	The visual representation of the 'axiomatic Claim' is considered feasible to be implemented in a tool support.
	Other constraints (if any)	N/A.
6	Compositional rules when used in a diagram	<p>An 'axiomatic Claim' can be used as a supporting 'Claim' that supported another 'Claim' or a particular type of 'AssertedRelationship type'. It can be placed below the supporting element if the 'axiomatic Claim' supporting another 'Claim', 'AssertedContext', or 'AssertedArtifactContext' relationship. In such cases, the relationship between the 'axiomatic Claim' and its targeted element must be declared using an 'AssertedInference' relationship. When an 'axiomatic Claim' is used as a 'Claim' that provides context to another 'Claim', 'AssertedInference', or 'AssertedEvidence' relationship, it must be located horizontally (left/right side) relative to its targeted elements. The relationship between the 'assumed Claim' and its targeted element must be declared using an 'AssertedContext' relationship.</p> <p>An 'axiomatic Claim' must not be supported by other elements in terms of elements that need to be attached below the 'axiomatic Claim' due to consider its semantics as a 'Claim' that is declared as axiomatically true, so that no further argumentation is needed.</p>

A.1.5 'defeated Claim'

TABLE A.5: 'defeated Claim' visual representation identity

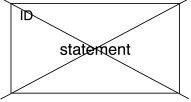
Visual representation identify of		'defeated Claim'
1	Visual representation	
2	Semantics	A 'Claim' that is defeated by counter argument/evidence.
3	Visual variable used	<p>Inherited from the design constraints of a 'Claim':</p> <ul style="list-style-type: none"> • Shape: 'Rectangle' • Brightness: 'Light' - background of the Rectangle. • Location: 'Vertical/horizontal' relative to its connected element; 'Top-Left' within the rectangle for the element ID (inherited from 'gid' attribute). • Texture: 'Solid line' - line texture of the Rectangle. • Size: "1pt" - the size of the line that form the rectangle.

Table A.5 continued from previous page

Visual representation identify of		'defeated Claim'
		Inherited from the 'defeated' design constraints: <ul style="list-style-type: none"> • Shape: 'A-double-lines' that form a cross symbol. • Location: The cross symbol is placed on top of the rectangle. • Size: '1 pt' line size. The width and height of the cross can be adjusted relative to the size of the rectangle. • Texture: 'Solid' for the texture of the line.
4	Visual inheritance info	Inherited design constraints of 'Claim', 'gid', and 'defeated'.
5	Design rationale	
	Avoiding clash with the existing notations	No existing notation in the same domain is used the visual representation of the 'defeated Claim'.
	Adopting existing visual notation design principles	
	Semiotic clarity	The 'defeated Claim' visual representation is only used for representing the 'defeated Claim'.
	Semantic transparency	Considered as an index visual representation for the user who already learn about the SACM 'Claim' and 'defeated'.
	Perceptual discriminability	The visual representation of the 'defeated Claim' can be differentiated with other created visual representation such as 'axiomatic Claim' and 'needsSupport Claim'.
	Redundant coding	The visual representation of the 'defeated Claim' is created using multiple visual variables.
	Notation Usability	
	Easy to draw	The visual representation of the 'defeated Claim' is considered relatively easy to draw.
	Use of colour	Not using Colour to create the visual representation of the 'defeated Claim'.
Tool support implementation	The visual representation of the 'defeated Claim' is considered feasible to be implemented in a tool support.	
Other constraints (if any)	N/A.	
6	Compositional rules when used in a diagram	The used of a 'defeated Claim' is to show a historical assurance case development that a particular 'Claim' has been asserted in a diagram, however, it is now being defeated by counter-argument/evidence. A 'defeated Claim' can be used as a supporting 'Claim' that supported another 'Claim' or a particular type of 'AssertedRelationship type'. It can be placed below the supporting element if the 'defeated Claim' supporting another 'Claim', 'AssertedContext', or 'AssertedArtifactContext' relationship. In such cases, the relationship between the 'defeated Claim' and its targeted element must be declared using an 'AssertedInference' relationship. When a 'defeated Claim' is is used as a 'Claim' that provides context to another 'Claim', 'AssertedInference', or 'AssertedEvidence' relationship, it must be located horizontally (left/right side) relative to its targeted elements. The relationship between the 'assumed Claim' and its targeted element must be declared using an 'AssertedContext' relationship. It is important to be noted that other elements that are countering the 'defeated Claim' must be presented to show that this claim is defeated by other argumentation/evidence.

A.1.6 'asCited Claim'

TABLE A.6: 'asCited Claim' visual representation identity

Visual representation identify of		'asCited Claim'
1	Visual representation	

Table A.6 continued from previous page

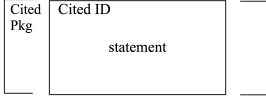
Visual representation identify of		'asCited Claim'
		
2	Semantics	A 'Claim' which cites another 'Claim'.
3	Visual variable used	<p>Inherited from the design constraints of a 'Claim':</p> <ul style="list-style-type: none"> • Shape: 'Rectangle' • Brightness: 'Light' - background of the Rectangle. • Location: 'Vertical/horizontal' relative to its connected element; 'Top-Left' within the rectangle for the cited element ID and 'Top-Left' within the square brackets for the location of the cited element (inherited from 'gid' attribute). • Texture: 'Solid line' - line texture of the Rectangle. • Size: "1pt" - the size of the line that form the rectangle.
		<p>Inherited from the 'asCited' design constraints:</p> <ul style="list-style-type: none"> • Shape: 'Lines' that form square brackets symbol. • Location: The 'Claim' must be located within the square brackets. The placement of the 'asCited Claim' in a diagram is following the compositional rules of the 'Claim'. • Size: '1 pt' line size that form the square brackets. • Texture: 'Solid' for the texture of the line.
4	Visual inheritance info	Inherited design constraints of 'Claim', 'gid', and 'asCited'.
5	Design rationale	
	Avoiding clash with the existing notations	No existing notation in the same domain is used the visual representation of the 'asCited Claim'.
	Adopting existing visual notation design principles	
	Semiotic clarity	The 'asCited Claim' visual representation is only used for representing the 'defeated Claim'.
	Semantic transparency	Considered as an index visual representation for the user who already learn about the SACM 'Claim' and 'asCited'.
	Perceptual discriminability	The visual representation of the 'asCited Claim' can be differentiated with other created visual representation such as 'defeated Claim' and 'needsSupport Claim'.
	Redundant coding	The visual representation of the 'asCited Claim' is created using multiple visual variables.
	Notation Usability	
	Easy to draw	The visual representation of the 'asCited Claim' is considered relatively easy to draw.
	Use of colour	Not using Colour to create the visual representation of the 'asCited Claim'.
Tool support implementation	The visual representation of the 'asCited Claim' is considered feasible to be implemented in a tool support.	
Other constraints (if any)		N/A.

Table A.6 continued from previous page

Visual representation identify of		‘asCited Claim’
6	Compositional rules when used in a diagram	<p>An ‘asCited Claim’ can be supported by another ‘Claim’ (connected via ‘AssertedInference’ relationship) or ‘ArtifactReference’ as a reference to evidential information (connected via ‘AssertedEvidence’ relationship). These supporting elements must be located below the ‘asCited Claim’.</p> <p>Another ‘Claim’ or ‘ArtifactReference’ can also be used to scope or provides context to an ‘asCited Claim’, in this case, these supporting elements must be located horizontally (left/right side) relative to the ‘asCited Claim’. The relationship between these elements must be declared using an ‘AssertedContext’ relationship.</p> <p>An ‘asCited Claim’ can be used as a supporting ‘Claim’ that supported another ‘Claim’ or a particular type of ‘AssertedRelationship type’. It can be placed below the supporting element if the ‘asCited Claim’ supporting another ‘Claim’, ‘AssertedContext’, or ‘AssertedArtifactContext’ relationship. In such cases, the relationship between the ‘defeated Claim’ and its targeted element must be declared using an ‘AssertedInference’ relationship.</p> <p>When an ‘asCited Claim’ is used as a ‘Claim’ that provides context to another ‘Claim’, ‘AssertedInference’, or ‘AssertedEvidence’ relationship, it must be located horizontally (left/right side) relative to its targeted elements. The relationship between the ‘asCited Claim’ and its targeted element must be declared using an ‘AssertedContext’ relationship.</p>

A.1.7 ‘abstract Claim’

TABLE A.7: ‘abstract Claim’ visual representation identity


Visual representation identify of		‘abstract Claim’
1	Visual representation	
2	Semantics	A ‘Claim’ that is declared as part of a pattern or template.
3	Visual variable used	<p>Inherited from the design constraints of a ‘Claim’:</p> <ul style="list-style-type: none"> • Shape: ‘Rectangle’ • Brightness: ‘Light’ - background of the Rectangle. • Location: ‘Vertical/horizontal’ relative to its connected element; ‘Top-Left’ within the rectangle for the element ID (inherited from ‘gid’ attribute). • Texture: ‘Dash line’ - line texture of the Rectangle (respecting the inherited ‘isAbstract’ design constraint). • Size: "1pt" - the size of the line that form the rectangle.
4	Visual inheritance info	Inherited design constraints of ‘Claim’, ‘gid’, and ‘abstract’.
5	Design rationale	
	Avoiding clash with the existing notations	No existing notation in the same domain is used the visual representation of the ‘abstract Claim’.
	Adopting existing visual notation design principles	
	Semiotic clarity	The ‘abstract Claim’ visual representation is only used for representing the ‘abstract Claim’.
Semantic transparency	Considered as an index visual representation for the user who already learn about the SACM ‘Claim’ and ‘abstract’.	

Table A.7 continued from previous page

Visual representation identify of		'abstract Claim'
	Perceptual discriminability	The visual representation of the 'abstract Claim' can be differentiated with other created visual representation such as 'defeated Claim' and 'asCited Claim'.
	Redundant coding	The visual representation of the 'abstract Claim' is created using multiple visual variables.
	Notation Usability	
	Easy to draw	The visual representation of the 'abstract Claim' is considered relatively easy to draw.
	Use of colour	Not using Colour to create the visual representation of the 'abstract Claim'.
	Tool support implementation	The visual representation of the 'abstract Claim' is considered feasible to be implemented in a tool support.
	Other constraints (if any)	N/A.
6	Compositional rules when used in a diagram	<p>An 'abstract Claim' can be used as a supporting 'Claim' to another 'Claim' and the relationship between them must be declared using an 'AssertedInference' relationship. The 'abstract Claim' must be located below the supported 'Claim'.</p> <p>An 'abstract Claim' can be connected to more than one 'abstract ArtifactReferences', in this case, the 'abstract ArtifactReferences' provides context to the 'abstract Claim'. The relationship between them can be declared using 'AssertedContext' relationship.</p>

A.2 'AssertedInference' and its variants

A.2.1 'AssertedInference'

TABLE A.8: 'AssertedInference' visual representation identity


Visual representation identify of		'AssertedInference'
1	Visual representation	
2	Semantics	'AssertedInference' association records the inference that a user declares to exist between one or more 'Assertion' (premise) and another 'Assertion' (conclusion).
3	Visual variable used	<ul style="list-style-type: none"> • Shape: 'Arrowhead-line' and 'Circle'. • Location: The edge without an arrow must be connected to the source element of the relationship. The edge with an arrow must be connected to the target element of the relationship. • Size: '1 pt' line size thickness for the line; '0.25' cm for the width and height of the circle (adjustable relative to the application in a diagram). • Texture: 'Solid' for the texture of the line. • Brightness: 'Dark' for the arrowhead and the circle.
4	Visual inheritance info	Inherited design constraints from (abstract) 'AssertedRelationship' class: must be drawn as a line with a 2D-type line-head as a pointer to indicates the target element of the relationship; and the other line-end (without line-head) must be used as a pointer to indicates the source element of the relationship.
	Design rationale	

Table A.8 continued from previous page

Visual representation identify of		'AssertedInference'
	Avoiding clash with the existing notations	Adopted from the GSN SupportedBy relationship with a dot added in the middle of the line as a connection point.
	Adopting existing visual notation design principles	
	Semiotic clarity	The visual representation of the 'AssertedInference' looks identical with the visual representation of the 'AssertedEvidence' and 'AssertedArtifactSupport' due to consider graphic complexity of a notation. They can be differentiated by identifying the source and target elements that are connected to them.
	Semantic transparency	Considered as an index visual representation in order to provide a clue to the users who familiar with the existing notation (GSN SupportedBy relationship).
	Perceptual discriminability	The visual representation of the 'AssertedInference' can be differentiated with other visual representations.
	Redundant coding	The visual representation of the 'AssertedInference' is created using multiple visual variables.
	Notation Usability	
	Easy to draw	The visual representation of the 'AssertedInference' is considered relatively easy to draw.
	Use of colour	Not using Colour to create the visual representation of the 'AssertedInference'.
	Tool support implementation	A dot that is placed in the middle of the line is added due to consider the tool support implementation factor (input from notation stakeholder). Therefore, the visual representation of the 'AssertedInference' is considered feasible to be implemented in a tool support.
	Other constraints (if any)	Considering graphic complexity (i.e. managing the number of the visual representations of a notation).
6	Compositional rules when used in a diagram	An 'AssertedInference' relationship must be drawn vertically relative to its connected elements. The supporting element as the source element of the relationship must be located below the supported element. An 'AssertedInference' can be connected to the following elements: <ul style="list-style-type: none"> • From (source element) : 'Claim' type • To (target element) : 'Assertion' type (e.g. 'Claim')

A.2.2 'needsSupport AssertedInference'

TABLE A.9: 'needsSupport AssertedInference' visual representation identity


Visual representation identify of		'needsSupport AssertedInference'
1	Visual representation	
2	Semantics	An 'AssertedInference' that is intentionally declared as requiring further evidence or argumentation
3	Visual variable used	Inherited from the design constraints of an 'AssertedInference': <ul style="list-style-type: none"> • Shape: 'Arrowhead-line'. • Location: The edge without an arrow must be connected to the source element of the relationship. The edge with an arrow must be connected to the target element of the relationship. • Size: '1 pt' line size thickness for the line. • Texture: 'Solid' for the texture of the line. • Brightness: 'Dark' for the arrowhead.

Table A.9 continued from previous page

Visual representation identify of		‘needsSupport AssertedInference’
		Inherited from the ‘needsSupport’ design constraints: <ul style="list-style-type: none"> • Shape: ‘3 Circles’ • Brightness: ‘Dark’ • Location: ‘on the middle of the line’ when being implemented in a ‘AssertedRelationship’ type. • Size: ‘1pt’ as the size of the line that form the Circles; ‘0.25’ cm for the width and height of the circles (adjustable relative to the application in a diagram).
4	Visual inheritance info	Inherited design constraints of ‘AssertedInference’ and ‘needsSupport’.
5	Design rationale	
	Avoiding clash with the existing notations	No existing notation in the same domain is used the visual representation of the ‘needsSupport AssertedInference’.
	Adopting existing visual notation design principles	
	Semiotic clarity	The visual representation of the ‘needsSupport AssertedInference’ looks identical with the visual representation of the ‘needsSupport AssertedEvidence’ and ‘needsSupport AssertedArtifactSupport’ due to consider graphic complexity of a notation. They can be differentiated by identifying the source and target elements that are connected to them.
	Semantic transparency	Considered as an index visual representation for the user who already learn about the SACM ‘AssertedInference’ and ‘needsSupport’.
	Perceptual discriminability	The visual representation of the ‘needsSupport AssertedInference’ can be differentiated with other created visual representation such as ‘AssertedInference’.
	Redundant coding	The visual representation of the ‘needsSupport AssertedInference’ is created using multiple visual variables.
	Notation Usability	
	Easy to draw	The visual representation of the ‘needsSupport AssertedInference’ is considered relatively easy to draw.
	Use of colour	Not using Colour to create the visual representation of the ‘needsSupport AssertedInference’.
Tool support implementation	The visual representation of the ‘needsSupport AssertedInference’ is considered feasible to be implemented in a tool support.	
Other constraints (if any)	Considering graphic complexity (i.e. managing the number of the visual representations of a notation).	
6	Compositional rules when used in a diagram	A ‘needsSupport AssertedInference’ relationship must be drawn vertically relative to its connected elements. The supporting element as the source element of the relationship must be located below the supported element. A ‘needsSupport AssertedInference’ can be connected to the following elements: <ul style="list-style-type: none"> • From (source element) : ‘Claim’ type • To (target element) : ‘Assertion’ type (e.g. ‘Claim’)

A.2.3 ‘assumed AssertedInference’

TABLE A.10: ‘assumed AssertedInference’ visual representation identity


Visual representation identify of		‘assumed AssertedInference’
1	Visual representation	

Table A.10 continued from previous page

Visual representation identify of		'assumed AssertedInference'
2	Semantics	An 'AssertedInference' that is intentionally declared without supporting evidence or argumentation.
3	Visual variable used	Inherited from the design constraints of an 'AssertedInference': <ul style="list-style-type: none"> • Shape: 'Arrowhead-line'. • Location: The edge without an arrow must be connected to the source element of the relationship. The edge with an arrow must be connected to the target element of the relationship. • Size: '1 pt' line size thickness for the line. • Texture: 'Solid' for the texture of the line. • Brightness: 'Dark' for the arrowhead.
		Inherited from the 'assumed' design constraints: <ul style="list-style-type: none"> • Shape: 'A line with a gap in the middle of the line (with two vertical lines at the line-end where the line gap is)' • Location: 'On the middle of the line'. • Size: '1pt' as the size of the line. • Texture: 'Solid' for the texture of the line.
4	Visual inheritance info	Inherited design constraints of 'AssertedInference' and 'assumed'.
5	Design rationale	
	Avoiding clash with the existing notations	No existing notation in the same domain is used the visual representation of the 'assumed AssertedInference'.
	Adopting existing visual notation design principles	
	Semiotic clarity	The visual representation of the 'assumed AssertedInference' looks identical with the visual representation of the 'assumed AssertedEvidence' and 'assumed AssertedArtifactSupport' due to consider graphic complexity of a notation. They can be differentiated by identifying the source and target elements that are connected to them.
	Semantic transparency	Considered as an index visual representation for the user who already learn about the SACM 'AssertedInference' and 'assumed'.
	Perceptual discriminability	The visual representation of the 'assumed AssertedInference' can be differentiated with other created visual representation such as 'AssertedInference'.
	Redundant coding	The visual representation of the 'assumed AssertedInference' is created using multiple visual variables.
	Notation Usability	
	Easy to draw	The visual representation of the 'assumed AssertedInference' is considered relatively easy to draw.
	Use of colour	Not using Colour to create the visual representation of the 'assumed AssertedInference'.
Tool support implementation	The visual representation of the 'assumed AssertedInference' is considered feasible to be implemented in a tool support.	
Other constraints (if any)	Considering graphic complexity (i.e. managing the number of the visual representations of a notation).	
6	Compositional rules when used in a diagram	An 'assumed AssertedInference' relationship must be drawn vertically relative to its connected elements. The supporting element as the source element of the relationship must be located below the supported element. An 'assumed AssertedInference' can be connected to the following elements: <ul style="list-style-type: none"> • From (source element) : 'Claim' type • To (target element) : 'Assertion' type (e.g. 'Claim')

A.2.4 'axiomatic AssertedInference'

TABLE A.11: 'axiomatic AssertedInference' visual representation identity


Visual representation identify of		'axiomatic AssertedInference'
1	Visual representation	
2	Semantics	An 'AssertedInference' that is intentionally declared as axiomatically true, so that no further argumentation is needed.
3	Visual variable used	Inherited from the design constraints of an 'AssertedInference': <ul style="list-style-type: none"> • Shape: 'Arrowhead-line'. • Location: The edge without an arrow must be connected to the source element of the relationship. The edge with an arrow must be connected to the target element of the relationship. • Size: '1 pt' line size thickness for the line. • Texture: 'Solid' for the texture of the line. • Brightness: 'Dark' for the arrowhead.
		Inherited from the 'axiomatic' design constraints: <ul style="list-style-type: none"> • Shape: 'A line' - must be located in the middle of the line and positioned vertically if the line of the 'AssertedRelationship' type drawn horizontally, and positioned horizontally if the line of the 'AssertedRelationship' type drawn vertically. • Location/Position: 'In the middle of the line'. • Size: '3pt' as the size of the line. • Texture: 'Solid' for the texture of the line.
4	Visual inheritance info	Inherited design constraints of 'AssertedInference' and 'axiomatic'.
5	Design rationale	
	Avoiding clash with the existing notations	No existing notation in the same domain is used the visual representation of the 'axiomatic AssertedInference'.
	Adopting existing visual notation design principles	
	Semiotic clarity	The visual representation of the 'axiomatic AssertedInference' looks identical with the visual representation of the 'axiomatic AssertedEvidence' and 'axiomatic AssertedArtifactSupport' due to consider graphic complexity of a notation. They can be differentiated by identifying the source and target elements that are connected to them.
	Semantic transparency	Considered as an index visual representation for the user who already learn about the SACM 'AssertedInference' and 'axiomatic'.
	Perceptual discriminability	The visual representation of the 'axiomatic AssertedInference' can be differentiated with other created visual representation such as 'AssertedInference'.
	Redundant coding	The visual representation of the 'axiomatic AssertedInference' is created using multiple visual variables.
	Notation Usability	
	Easy to draw	The visual representation of the 'axiomatic AssertedInference' is considered relatively easy to draw.
	Use of colour	Not using Colour to create the visual representation of the 'axiomatic AssertedInference'.
Tool support implementation	The visual representation of the 'axiomatic AssertedInference' is considered feasible to be implemented in a tool support.	
Other constraints (if any)	Considering graphic complexity (i.e. managing the number of the visual representations of a notation).	

Table A.11 continued from previous page

Visual representation identify of		‘axiomatic AssertedInference’
6	Compositional rules when used in a diagram	<p>An ‘axiomatic AssertedInference’ relationship must be drawn vertically relative to its connected elements. The supporting element as the source element of the relationship must be located below the supported element.</p> <p>An ‘axiomatic AssertedInference’ can be connected to the following elements:</p> <ul style="list-style-type: none"> • From (source element) : ‘Claim’ type • To (target element) : ‘Assertion’ type (e.g. ‘Claim’)

A.2.5 ‘defeated AssertedInference’

TABLE A.12: ‘defeated AssertedInference’ visual representation identity


Visual representation identify of		‘defeated AssertedInference’
1	Visual representation	
2	Semantics	An ‘AssertedInference’ that is defeated by counter argument/evidence.
3	Visual variable used	<p>Inherited from the design constraints of an ‘AssertedInference’:</p> <ul style="list-style-type: none"> • Shape: ‘Arrowhead-line’. • Location/Position: The edge without an arrow must be connected to the source element of the relationship. The edge with an arrow must be connected to the target element of the relationship. • Size: ‘1 pt’ line size thickness for the line. • Texture: ‘Solid’ for the texture of the line. • Brightness: ‘Dark’ for the arrowhead.
		<p>Inherited from the ‘defeated’ design constraints:</p> <ul style="list-style-type: none"> • Shape: ‘A-double-lines’ that form a cross symbol. • Location: The cross symbol is placed in the middle of the line. • Size: ‘1 pt’ line size thickness for the lines that form a cross symbol; The height of cross must be drawn relatively in the size of the line-head of the relationship. • Texture: ‘Solid’ for the texture of the line.
4	Visual inheritance info	Inherited design constraints of ‘AssertedInference’ and ‘defeated’.
5	Design rationale	
	Avoiding clash with the existing notations	No existing notation in the same domain is used the visual representation of the ‘defeated AssertedInference’.
	Adopting existing visual notation design principles	
	Semiotic clarity	The visual representation of the ‘defeated AssertedInference’ looks identical with the visual representation of the ‘defeated AssertedEvidence’ and ‘defeated AssertedArtifactSupport’ due to consider graphic complexity of a notation. They can be differentiated by identifying the source and target elements that are connected to them.
	Semantic transparency	Considered as an index visual representation for the user who already learn about the SACM ‘AssertedInference’ and ‘defeated’.
	Perceptual discriminability	The visual representation of the ‘defeated AssertedInference’ can be differentiated with other created visual representation such as ‘AssertedInference’.
Redundant coding	The visual representation of the ‘defeated AssertedInference’ is created using multiple visual variables.	
Notation Usability		

Table A.12 continued from previous page

Visual representation identify of		‘defeated AssertedInference’
	Easy to draw	The visual representation of the ‘defeated AssertedInference’ is considered relatively easy to draw.
	Use of colour	Not using Colour to create the visual representation of the ‘defeated AssertedInference’.
	Tool support implementation	The visual representation of the ‘defeated AssertedInference’ is considered feasible to be implemented in a tool support.
	Other constraints (if any)	Considering graphic complexity (i.e. managing the number of the visual representations of a notation).
6	Compositional rules when used in a diagram	<p>A ‘defeated AssertedInference’ relationship must be drawn vertically relative to its connected elements. The supporting element as the source element of the relationship must be located below the supported element.</p> <p>An ‘defeated AssertedInference’ can be connected to the following elements:</p> <ul style="list-style-type: none"> • From (source element) : ‘Claim’ type • To (target element) : ‘Assertion’ type (e.g. ‘Claim’)

A.2.6 ‘asCited AssertedInference’

TABLE A.13: ‘asCited AssertedInference’ visual representation identity


Visual representation identify of		‘asCited AssertedInference’
1	Visual representation	
2	Semantics	An ‘AssertedInference’ which cites another ‘AssertedInference’.
3	Visual variable used	<p>Inherited from the design constraints of an ‘AssertedInference’:</p> <ul style="list-style-type: none"> • Shape: ‘Arrowhead-line’ and ‘Circle’. • Location/Position: The edge without an arrow must be connected to the source element of the relationship. The edge with an arrow must be connected to the target element of the relationship. • Size: ‘1 pt’ line size thickness for the line. • Texture: ‘Solid’ for the texture of the line. • Brightness: ‘Dark’ for the arrowhead and the circle.
		<p>Inherited from the ‘asCited’ design constraints:</p> <ul style="list-style-type: none"> • Shape: ‘Lines’ that form square brackets symbol (‘[]’). • Location/Position: The ‘AssertedInference’ must be located within the square brackets. • Size: ‘1 pt’ line size thickness for the lines that form the square brackets. • Texture: ‘Solid’ for the texture of the line.
4	Visual inheritance info	Inherited design constraints of ‘AssertedInference’ and ‘asCited’.
5	Design rationale	
	Avoiding clash with the existing notations	No existing notation in the same domain is used the visual representation of the ‘asCited AssertedInference’.
	Adopting existing visual notation design principles	
	Semiotic clarity	The visual representation of the ‘asCited AssertedInference’ looks identical with the visual representation of the ‘asCited AssertedEvidence’ and ‘asCited AssertedArtifactSupport’ due to consider graphic complexity of a notation. They can be differentiated by identifying the source and target elements that are connected to them.

Table A.13 continued from previous page

Visual representation identify of		‘asCited AssertedInference’
	Semantic transparency	Considered as an index visual representation for the user who already learn about the SACM ‘AssertedInference’ and ‘asCited’.
	Perceptual discriminability	The visual representation of the ‘asCited AssertedInference’ can be differentiated with other created visual representation such as ‘AssertedInference’.
	Redundant coding	The visual representation of the ‘asCited AssertedInference’ is created using multiple visual variables.
	Notation Usability	
	Easy to draw	The visual representation of the ‘asCited AssertedInference’ is considered relatively easy to draw.
	Use of colour	Not using Colour to create the visual representation of the ‘asCited AssertedInference’.
	Tool support implementation	The visual representation of the ‘asCited AssertedInference’ is considered feasible to be implemented in a tool support.
	Other constraints (if any)	Considering graphic complexity (i.e. managing the number of the visual representations of a notation).
6	Compositional rules when used in a diagram	An ‘asCited AssertedInference’ relationship must be drawn vertically relative to its connected elements. The supporting element as the source element of the relationship must be located below the supported element. An ‘asCited AssertedInference’ can be connected to the following elements: <ul style="list-style-type: none"> • From (source element) : ‘Claim’ type • To (target element) : ‘Assertion’ type (e.g. ‘Claim’)

A.2.7 ‘abstract AssertedInference’

TABLE A.14: ‘abstract AssertedInference’ visual representation identity

Visual representation identify of		‘abstract AssertedInference’
1	Visual representation	-----▶
2	Semantics	An ‘AssertedInference’ that is declared as part of a pattern or template.
3	Visual variable used	Inherited from the design constraints of an ‘AssertedInference’: <ul style="list-style-type: none"> • Shape: ‘Arrowhead-line’. • Location/Position: The edge without an arrow must be connected to the source element of the relationship. The edge with an arrow must be connected to the target element of the relationship. • Size: ‘1 pt’ line size thickness for the line. • Texture: ‘Dash line’ - line texture of the Rectangle (respecting the inherited ‘isAbstract’ design constraint). • Brightness: ‘Dark’ for the arrowhead.
4	Visual inheritance info	Inherited design constraints of ‘AssertedInference’ and ‘isAbstract’.
	Design rationale	
	Avoiding clash with the existing notations	No existing notation in the same domain is used the visual representation of the ‘abstract AssertedInference’.
	Adopting existing visual notation design principles	
5	Semiotic clarity	The visual representation of the ‘abstract AssertedInference’ looks identical with the visual representation of the ‘abstract AssertedEvidence’ and ‘abstract AssertedArtifactSupport’ due to consider graphic complexity of a notation. They can be differentiated by identifying the source and target elements that are connected to them.

Table A.14 continued from previous page

Visual representation identify of		'abstract AssertedInference'
	Semantic transparency	Considered as an index visual representation for the user who already learn about the SACM 'AssertedInference' and 'abstract'.
	Perceptual discriminability	The visual representation of the 'abstract AssertedInference' can be differentiated with other created visual representation such as 'AssertedInference'.
	Redundant coding	The visual representation of the 'abstract AssertedInference' is created using multiple visual variables.
	Notation Usability	
	Easy to draw	The visual representation of the 'abstract AssertedInference' is considered relatively easy to draw.
	Use of colour	Not using Colour to create the visual representation of the 'abstract AssertedInference'.
	Tool support implementation	The visual representation of the 'abstract AssertedInference' is considered feasible to be implemented in a tool support.
	Other constraints (if any)	Considering graphic complexity (i.e. managing the number of the visual representations of a notation).
6	Compositional rules when used in a diagram	An 'abstract AssertedInference' relationship must be drawn vertically relative to its connected elements. The supporting element as the source element of the relationship must be located below the supported element. An 'abstract AssertedInference' can be connected to the following elements: <ul style="list-style-type: none"> • From (source element) : 'Claim' type • To (target element) : 'Assertion' type (e.g. 'Claim')

A.2.8 'counter AssertedInference'

TABLE A.15: 'counter AssertedInference' visual representation identity

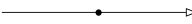
Visual representation identify of		'counter AssertedInference'
1	Visual representation	
2	Semantics	An 'AssertedInference' that counters its declared purposes.
3	Visual variable used	Inherited from the design constraints of an 'AssertedInference': <ul style="list-style-type: none"> • Shape: 'Arrowhead-line' and 'Circle'. • Location/Position: The edge without an arrow must be connected to the source element of the relationship. The edge with an arrow must be connected to the target element of the relationship. • Size: '1 pt' line size thickness for the line. • Texture: 'Solid' for the texture of the line. • Brightness: 'Light' for the arrowhead (applying the inherited design constraint of the 'isCounter'); and 'dark' for the circle.
4	Visual inheritance info	Inherited design constraints of 'AssertedInference' and 'isCounter'.
	Design rationale	
	Avoiding clash with the existing notations	No existing notation in the same domain is used the visual representation of the 'counter AssertedInference'.
	Adopting existing visual notation design principles	
5	Semiotic clarity	The visual representation of the 'counter AssertedInference' looks identical with the visual representation of the 'counter AssertedEvidence' and 'counter AssertedArtifactSupport' due to consider graphic complexity of a notation. They can be differentiated by identifying the source and target elements that are connected to them.
	Semantic transparency	Considered as an index visual representation for the user who already learn about the SACM 'AssertedInference' and 'abstract'.

Table A.15 continued from previous page

Visual representation identify of		‘counter AssertedInference’
	Perceptual discriminability	The visual representation of the ‘counter AssertedInference’ can be differentiated with other created visual representation such as ‘AssertedInference’.
	Redundant coding	The visual representation of the ‘counter AssertedInference’ is created using multiple visual variables.
	Notation Usability	
	Easy to draw	The visual representation of the ‘counter AssertedInference’ is considered relatively easy to draw.
	Use of colour	Not using Colour to create the visual representation of the ‘counter AssertedInference’.
	Tool support implementation	The visual representation of the ‘counter AssertedInference’ is considered feasible to be implemented in a tool support.
	Other constraints (if any)	Considering graphic complexity (i.e. managing the number of the visual representations of a notation).
6	Compositional rules when used in a diagram	<p>A ‘counter AssertedInference’ relationship must be drawn vertically relative to its connected elements. The supporting element as the source element of the relationship must be located below the supported element.</p> <p>A ‘counter AssertedInference’ can be connected to the following elements:</p> <ul style="list-style-type: none"> • From (source element) : ‘Claim’ type • To (target element) : ‘Assertion’ type (e.g. ‘Claim’)

A.3 ‘ArtifactReference’ and its variants

A.3.1 ‘ArtifactReference’

TABLE A.16: ‘ArtifactReference’ visual representation identity

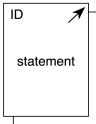
Visual representation identify of		‘ArtifactReference’
1	Visual representation	
2	Semantics	It is necessary to be able to cite artifacts that provide supporting evidence, context, or additional description with in an argument structure. ‘ArtifactReferences’ allow there to be an objectified citation of this information within the structured argument, thereby allowing the relationship between this artifact and the argument to also be explicitly declared.

Table A.16 continued from previous page

Visual representation identify of		'ArtifactReference'
3	Visual variable used	<ul style="list-style-type: none"> • Shape: '2 (vertical) Rectangles' and '1D arrowhead-line' combined together forming a document icon. • Location: 'Both rectangles placed vertically with one rectangle must be located on top of another rectangle. The right and bottom part of the rectangle that is placed below another rectangle must be slightly appeared (as shown in the figure of 'ArtifactReference' visual representation). An arrowhead-line must be placed on the right-top corner within the top rectangle'; 'Top-Left' within the rectangle for the element ID (inherited from 'gid' attribute). • Size: '1 pt' line size thickness. • Texture: 'Solid' for the texture of the line. • Brightness: 'Bright' as the background of the 'ArtifactReference'.
4	Visual inheritance info	Inherited design constraints of 'gid'.
5	Design rationale	
	Avoiding clash with the existing notations	No existing notation in the same domain is used the visual representation of the 'ArtifactReference'.
	Adopting existing visual notation design principles	
	Semiotic clarity	The visual representation of the 'ArtifactReference' is only used for the visual representation of the 'ArtifactReference'.
	Semantic transparency	The 'ArtifactReference' visual representation is considered as an icon-type visual representation because its appearance may suggests or represents its semantic meaning.
	Perceptual discriminability	The visual representation of the 'ArtifactReference' can be differentiated with other created visual representation such as 'Claim'.
	Redundant coding	The visual representation of the 'ArtifactReference' is created using multiple visual variables.
	Notation Usability	
Easy to draw	The visual representation of the 'ArtifactReference' is considered relatively easy to draw.	
Use of colour	Not using Colour to create the visual representation of the 'ArtifactReference'.	
Tool support implementation	The visual representation of the 'ArtifactReference' is considered feasible to be implemented in a tool support.	
Other constraints (if any)	N/A.	
6	Compositional rules when used in a diagram	<p>When an 'ArtifactReference' is used as a reference to an evidential information, it must be located below the target element:</p> <ul style="list-style-type: none"> • If the target element is a 'Claim', the relationship between them must be declared using an 'AssertedEvidence'. • If the target element is another 'ArtifactReference' (as a reference to context), the relationship between them must be declared using an 'AssertedArtifactSupport'. <p>When an 'ArtifactReference' is used as a reference to a contextual information, it must be located horizontally (right/left side) relative to its target element:</p> <ul style="list-style-type: none"> • If the target element is a 'Claim', the relationship between them must be declared using an 'AssertedContext'. • If the target element is another 'ArtifactReference' (as a reference to evidence), the relationship between them must be declared using an 'AssertedArtifactContext'.

A.3.2 'abstract ArtifactReference'

TABLE A.17: 'abstract ArtifactReference' visual representation identity

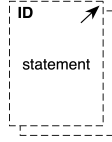
Visual representation identify of		'abstract ArtifactReference'
1	Visual representation	
2	Semantics	An 'ArtifactReference' that is declared as part of pattern or template.
3	Visual variable used	<p>Inherited from the design constraints of an 'ArtifactReference':</p> <ul style="list-style-type: none"> • Shape: '2 (vertical) Rectangles' and '1D arrowhead-line' combined together forming a document icon. • Location: 'Both rectangles placed vertically with one rectangle must be located on top of another rectangle. The right and bottom part of the rectangle that is placed below another rectangle must be slightly appeared (as shown in the figure of 'ArtifactReference' visual representation). An arrowhead-line must be placed on the right-top corner within the top rectangle'; 'Top-Left' within the rectangle for the element ID (inherited from 'gid' attribute). • Size: '1 pt' line size thickness. • Texture: 'Dash line' - line texture of the Rectangle (respecting the inherited 'isAbstract' design constraint). • Brightness: 'Bright' as the background of the 'abstract ArtifactReference'.
4	Visual inheritance info	Inherited design constraints of 'ArtifactReference', 'gid', and 'isAbstract'.
5	Design rationale	
	Avoiding clash with the existing notations	No existing notation in the same domain is used the visual representation of the 'abstract ArtifactReference'.
	Adopting existing visual notation design principles	
	Semiotic clarity	The visual representation of the 'abstract ArtifactReference' is only used for the visual representation of the 'abstract ArtifactReference'.
	Semantic transparency	The 'abstract ArtifactReference' visual representation is considered as an index-type visual representation for the notation user who have learned about SACM 'ArtifactReference' and 'abstract' visual representation.
	Perceptual discriminability	The visual representation of the 'abstract ArtifactReference' can be differentiated with other created visual representation such as 'ArtifactReference'.
	Redundant coding	The visual representation of the 'abstract ArtifactReference' is created using multiple visual variables.
	Notation Usability	
	Easy to draw	The visual representation of the 'abstract ArtifactReference' is considered relatively easy to draw.
	Use of colour	Not using Colour to create the visual representation of the 'abstract ArtifactReference'.
Tool support implementation	The visual representation of the 'abstract ArtifactReference' is considered feasible to be implemented in a tool support.	
Other constraints (if any)		N/A.

Table A.17 continued from previous page

Visual representation identify of		‘abstract ArtifactReference’
6	Compositional rules when used in a diagram	<p>When an ‘abstract ArtifactReference’ is used as a reference to an evidential information, it must be located below the target element:</p> <ul style="list-style-type: none"> • If the target element is a ‘Claim’, the relationship between them must be declared using an ‘AssertedEvidence’. • If the target element is another ‘ArtifactReference’ (as a reference to context), the relationship between them must be declared using an ‘AssertedArtifactSupport’. <p>When an ‘abstract ArtifactReference’ is used as a reference to a contextual information, it must be located horizontally (right/left side) relative to its target element:</p> <ul style="list-style-type: none"> • If the target element is a ‘Claim’, the relationship between them must be declared using an ‘AssertedContext’. • If the target element is another ‘ArtifactReference’ (as a reference to evidence), the relationship between them must be declared using an ‘AssertedArtifactContext’.

A.3.3 ‘asCited ArtifactReference’

TABLE A.18: ‘asCited ArtifactReference’ visual representation identity

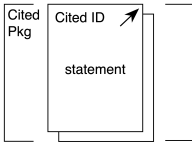
Visual representation identify of		‘asCited ArtifactReference’
1	Visual representation	
2	Semantics	An ‘ArtifactReference’ which cites another ‘ArtifactReference’.
3	Visual variable used	<p>Inherited from the design constraints of an ‘ArtifactReference’:</p> <ul style="list-style-type: none"> • Shape: ‘2 (vertical) Rectangles’ and ‘1D arrowhead-line’ combined together forming a document icon. • Location: ‘Both rectangles placed vertically with one rectangle must be located on top of another rectangle. The right and bottom part of the rectangle that is placed below another rectangle must be slightly appeared (as shown in the figure of ‘ArtifactReference’ visual representation). An arrowhead-line must be placed on the right-top corner within the top rectangle’; ‘Top-Left’ within the rectangle for the element ID (inherited from ‘gid’ attribute). • Size: ‘1 pt’ line size thickness. • Texture: ‘Solid’ for the texture of the line. • Brightness: ‘Bright’ as the background of the ‘ArtifactReference’. <p>Inherited from the ‘asCited’ design constraints:</p> <ul style="list-style-type: none"> • Shape: ‘Lines’ that form square brackets symbol. • Location: The ‘ArtifactReference’ must be located within the square brackets. • Size: ‘1 pt’ line size that form the square brackets. • Texture: ‘Solid’ for the texture of the line.
4	Visual inheritance info	Inherited design constraints of ‘ArtifactReference’, ‘gid’, and ‘asCited’.
	Design rationale	

Table A.18 continued from previous page

	Visual representation identify of	‘asCited ArtifactReference’
	Avoiding clash with the existing notations	No existing notation in the same domain is used the visual representation of the ‘asCited ArtifactReference’.
	Adopting existing visual notation design principles	
	Semiotic clarity	The visual representation of the ‘asCited ArtifactReference’ is only used for the visual representation of the ‘asCited ArtifactReference’.
	Semantic transparency	The ‘asCited ArtifactReference’ visual representation is considered as an index-type visual representation for the notation user who have learned about SACM ‘ArtifactReference’ and ‘asCited’ visual representation.
	Perceptual discriminability	The visual representation of the ‘asCited ArtifactReference’ can be differentiated with other created visual representation such as ‘ArtifactReference’.
	Redundant coding	The visual representation of the ‘asCited ArtifactReference’ is created using multiple visual variables.
	Notation Usability	
	Easy to draw	The visual representation of the ‘asCited ArtifactReference’ is considered relatively easy to draw.
	Use of colour	Not using Colour to create the visual representation of the ‘asCited ArtifactReference’.
	Tool support implementation	The visual representation of the ‘asCited ArtifactReference’ is considered feasible to be implemented in a tool support.
	Other constraints (if any)	N/A.
6	Compositional rules when used in a diagram	<p>When an ‘asCited ArtifactReference’ is used as a reference to an evidential information, it must be located below the target element:</p> <ul style="list-style-type: none"> • If the target element is a ‘Claim’, the relationship between them must be declared using an ‘AssertedEvidence’. • If the target element is another ‘ArtifactReference’ (as a reference to context), the relationship between them must be declared using an ‘AssertedArtifactSupport’. <p>When an ‘asCited ArtifactReference’ is used as a reference to a contextual information, it must be located horizontally (right/left side) relative to its target element:</p> <ul style="list-style-type: none"> • If the target element is a ‘Claim’, the relationship between them must be declared using an ‘AssertedContext’. • If the target element is another ‘ArtifactReference’ (as a reference to evidence), the relationship between them must be declared using an ‘AssertedArtifactContext’.

A.4 ‘AssertedEvidence’ and its variants

Table A.19 presents the visual representation identity of ‘AssertedEvidence’. The other types of ‘AssertedEvidence’ element are created based on visual inheritance combining the design constraints of the ‘AssertedEvidence’ and the design constraints of a particular inherited attribute. For example, ‘defeated AssertedEvidence’ element is created by combining design constraints of the ‘AssertedEvidence’ and ‘defeated’. The illustration of the visual representation identity of the other types of ‘AssertedEvidence’ element is similar with the visual representation identity of the other types of ‘AssertedInference’ element.

A.4.1 ‘AssertedEvidence’

TABLE A.19: 'AssertedEvidence' visual representation identity

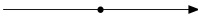
Visual representation identify of		'AssertedEvidence'
1	Visual representation	
2	Semantics	'AssertedEvidence' can be used to records the declaration that one or more artifacts of Evidence (cited by 'ArtifactReference') provide information that helps establish the truth of a 'Claim'.
3	Visual variable used	<ul style="list-style-type: none"> • Shape: 'Arrowhead-line' and 'Circle'. • Location: The edge without an arrow must be connected to the source element of the relationship (: 'ArtifactReference' as a reference to an evidential information). The edge with an arrow must be connected to the target element of the relationship (: 'Claim'). • Size: '1 pt' line size thickness for the line; '0.25' cm for the width and height of the circle (adjustable relative to the application in a diagram). • Texture: 'Solid' for the texture of the line. • Brightness: 'Dark' for the arrowhead and the circle.
4	Visual inheritance info	Inherited design constraints from (abstract) 'AssertedRelationship' class: must be drawn as a line with a 2D-type line-head as a pointer to indicates the target element of the relationship; and the other line-end (without line-head) must be used as a pointer to indicates the source element of the relationship.
5	Design rationale	
	Avoiding clash with the existing notations	Adopted from the GSN SupportedBy relationship with a dot added in the middle of the line as a connection point.
	Adopting existing visual notation design principles	
	Semiotic clarity	The visual representation of the 'AssertedEvidence' looks identical with the visual representation of the 'AssertedInference' and 'AssertedArtifactSupport' due to consider graphic complexity of a notation. They can be differentiated by identifying the source and target elements that are connected to them.
	Semantic transparency	Considered as an index visual representation in order to provide a clue to the users who familiar with the existing notation (GSN SupportedBy relationship).
	Perceptual discriminability	The visual representation of the 'AssertedEvidence' can be differentiated with other visual representations.
	Redundant coding	The visual representation of the 'AssertedEvidence' is created using multiple visual variables.
	Notation Usability	
	Easy to draw	The visual representation of the 'AssertedEvidence' is considered relatively easy to draw.
	Use of colour	Not using Colour to create the visual representation of the 'AssertedEvidence'.
Tool support implementation	A dot that is placed in the middle of the line is added due to consider the tool support implementation factor (input from notation stakeholder). Therefore, the visual representation of the 'AssertedEvidence' is considered feasible to be implemented in a tool support.	
Other constraints (if any)	Considering graphic complexity (i.e. managing the number of the visual representations of a notation).	

Table A.19 continued from previous page

Visual representation identify of		‘AssertedEvidence’
6	Compositional rules when used in a diagram	<p>An ‘AssertedEvidence’ relationship must be drawn vertically relative to its connected elements. The supporting element as the source element of the relationship must be located below the supported element.</p> <p>An ‘AssertedEvidence’ can be connected to the following elements:</p> <ul style="list-style-type: none"> • From (source element) : ‘ArtifactReference’ as a reference to an evidential information. • To (target element) : ‘Claim’.

A.5 ‘AssertedContext’ and its variants

A.5.1 ‘AssertedContext’

TABLE A.20: ‘AssertedContext’ visual representation identity

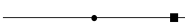
Visual representation identify of		‘AssertedContext’
1	Visual representation	
2	Semantics	<p>‘AssertedContext’ can be used to declare that the artifact cited by an ‘ArtifactReference’ provides the context for the interpretation and scoping of a ‘Claim’ element. It also can be used to declare a ‘Claim’ asserted as necessary context (i.e. a precondition) for another ‘Assertion’ such as a ‘Claim’.</p>
3	Visual variable used	<ul style="list-style-type: none"> • Shape: ‘A line’, ‘a square’ and ‘a Circle’. • Location/Position: The square must be located near one of the line-end that is used as a pointer to the targeted element. The circle must be located in the middle of the line. The other line-end is used to indicates the source element of the relationship. When used in the diagram the ‘AssertedContext’ must be drawn horizontally relative to its source and target elements. • Size: ‘1 pt’ line size thickness for the line; ‘0.25’ cm for the width and height of the circle; ‘0.30’ cm for the width and height of the square (adjustable relative to the application in a diagram). • Texture: ‘Solid’ for the texture of the line. • Brightness: ‘Dark’ for the square and the circle.
4	Visual inheritance info	<p>Inherited design constraints from (abstract) ‘AssertedRelationship’ class: must be drawn as a line with a 2D-type line-head as a pointer to indicates the target element of the relationship; and the other line-end (without line-head) must be used as a pointer to indicates the source element of the relationship.</p>
Design rationale		
5	Avoiding clash with the existing notations	No existing notation in the same domain is used the visual representation of the ‘AssertedContext’.
	Adopting existing visual notation design principles	
	Semiotic clarity	The visual representation of the ‘AssertedContext’ looks identical with the visual representation of the ‘AssertedArtifactContext’ due to consider graphic complexity of a notation. They can be differentiated by identifying the source and target elements that are connected to them.

Table A.20 continued from previous page

Visual representation identify of		'AssertedContext'
	Semantic transparency	The 'AssertedContext' relationship is considered as a symbol-type visual representation. The user needs to learn in order to understand the meaning of it.
	Perceptual discriminability	The visual representation of the 'AssertedContext' can be differentiated with other visual representations.
	Redundant coding	The visual representation of the 'AssertedContext' is created using multiple visual variables.
	Notation Usability	
	Easy to draw	The visual representation of the 'AssertedContext' is considered relatively easy to draw.
	Use of colour	Not using Colour to create the visual representation of the 'AssertedContext'.
	Tool support implementation	A dot that is placed in the middle of the line is added due to consider the tool support implementation factor (input from notation stakeholder). Therefore, the visual representation of the 'AssertedInference' is considered feasible to be implemented in a tool support.
	Other constraints (if any)	Considering graphic complexity (i.e. managing the number of the visual representations of a notation).
6	Compositional rules when used in a diagram	An 'AssertedContext' relationship must be drawn horizontally relative to its connected elements. Permitted connections of the 'AssertedContext': <ul style="list-style-type: none"> From (source element) : 'ArtifactReference' (as a Context) or a 'Claim' element (as a necessary context (i.e. a precondition)) To (target element) : 'Assertion' type (e.g. 'Claim').

A.5.2 'needsSupport AssertedContext'

TABLE A.21: 'needsSupport AssertedContext' visual representation identity


Visual representation identify of		'needsSupport AssertedContext'
1	Visual representation	
2	Semantics	An 'AssertedContext' that is intentionally declared as requiring further evidence or argumentation
3	Visual variable used	Inherited from the design constraints of an 'AssertedContext': <ul style="list-style-type: none"> Shape: 'A line' and 'a square'. Location/Position: The square must be located near one of the line-end that is used as a pointer to the targeted element. The other line-end is used to indicates the source element of the relationship. When used in the diagram the 'AssertedContext' must be drawn horizontally relative to its source and target elements. Size: '1 pt' line size thickness for the line; '0.30' cm for the width and height of the square (adjustable relative to the application in a diagram). Texture: 'Solid' for the texture of the line. Brightness: 'Dark' for the square.

Table A.21 continued from previous page

Visual representation identify of		‘needsSupport AssertedContext’
		Inherited from the ‘needsSupport’ design constraints: <ul style="list-style-type: none"> • Shape: ‘3 Circles’ • Brightness: ‘Dark’ • Location: ‘on the middle of the line’ when being implemented in a ‘AssertedRelationship’ type. • Size: ‘1pt’ as the size of the line that form the Circles; ‘0.25’ cm for the width and height of the circles (adjustable relative to the application in a diagram).
4	Visual inheritance info	Inherited design constraints of ‘AssertedContext’ and ‘needsSupport’.
5	Design rationale	
	Avoiding clash with the existing notations	No existing notation in the same domain is used the visual representation of the ‘needsSupport AssertedContext’.
	Adopting existing visual notation design principles	
	Semiotic clarity	The visual representation of the ‘needsSupport AssertedContext’ looks identical with the visual representation of the ‘needsSupport AssertedArtifact-Context’ due to consider graphic complexity of a notation. They can be differentiated by identifying the source and target elements that are connected to them.
	Semantic transparency	Considered as an index visual representation for the user who already learn about the SACM ‘AssertedContext’ and ‘needsSupport’.
	Perceptual discriminability	The visual representation of the ‘needsSupport AssertedContext’ can be differentiated with other visual representations.
	Redundant coding	The visual representation of the ‘needsSupport AssertedContext’ is created using multiple visual variables.
	Notation Usability	
	Easy to draw	The visual representation of the ‘needsSupport AssertedContext’ is considered relatively easy to draw.
	Use of colour	Not using Colour to create the visual representation of the ‘needsSupport AssertedContext’.
Tool support implementation	The visual representation of the ‘needsSupport AssertedContext’ is considered feasible to be implemented in a tool support.	
Other constraints (if any)	Considering graphic complexity (i.e. managing the number of the visual representations of a notation).	
6	Compositional rules when used in a diagram	A ‘needsSupport AssertedContext’ relationship must be drawn horizontally relative to its connected elements. Permitted connections of the ‘needsSupport AssertedContext’ (inherited from the ‘AssertedContext’): <ul style="list-style-type: none"> • From (source element) : ‘ArtifactReference’ (as a Context) or a ‘Claim’ element (as a necessary context (i.e. a precondition)) • To (target element) : ‘Assertion’ type (e.g. ‘Claim’).

A.5.3 ‘assumed AssertedContext’

TABLE A.22: ‘assumed AssertedContext’ visual representation identity


Visual representation identify of		‘assumed AssertedContext’
1	Visual representation	
2	Semantics	An ‘AssertedContext’ that is intentionally declared without supporting evidence or argumentation.

Table A.22 continued from previous page

	Visual representation identify of	'assumed AssertedContext'
3	Visual variable used	<p>Inherited from the design constraints of an 'AssertedContext':</p> <ul style="list-style-type: none"> • Shape: 'A line' and 'a square'. • Location/Position: The square must be located near one of the line-end that is used as a pointer to the targeted element. The other line-end is used to indicate the source element of the relationship. When used in the diagram the 'AssertedContext' must be drawn horizontally relative to its source and target elements. • Size: '1 pt' line size thickness for the line; '0.30' cm for the width and height of the square (adjustable relative to the application in a diagram). • Texture: 'Solid' for the texture of the line. • Brightness: 'Dark' for the square. <p>Inherited from the 'assumed' design constraints:</p> <ul style="list-style-type: none"> • Shape: 'A line with a gap in the middle of the line (with two vertical lines at the line-end where the line gap is)' • Location: 'On the middle of the line'. • Size: '1pt' as the size of the line. • Texture: 'Solid' for the texture of the line.
4	Visual inheritance info	Inherited design constraints of 'AssertedContext' and 'assumed'.
5	Design rationale	
	Avoiding clash with the existing notations	No existing notation in the same domain is used the visual representation of the 'assumed AssertedContext'.
	Adopting existing visual notation design principles	
	Semiotic clarity	The visual representation of the 'assumed AssertedContext' looks identical with the visual representation of the 'assumed AssertedArtifactContext' due to consider graphic complexity of a notation. They can be differentiated by identifying the source and target elements that are connected to them.
	Semantic transparency	Considered as an index visual representation for the user who already learn about the SACM 'AssertedContext' and 'assumed'.
	Perceptual discriminability	The visual representation of the 'assumed AssertedContext' can be differentiated with other visual representations.
	Redundant coding	The visual representation of the 'assumed AssertedContext' is created using multiple visual variables.
	Notation Usability	
Easy to draw	The visual representation of the 'assumed AssertedContext' is considered relatively easy to draw.	
Use of colour	Not using Colour to create the visual representation of the 'assumed AssertedContext'.	
Tool support implementation	The visual representation of the 'assumed AssertedContext' is considered feasible to be implemented in a tool support.	
Other constraints (if any)	Considering graphic complexity (i.e. managing the number of the visual representations of a notation).	
6	Compositional rules when used in a diagram	<p>An 'assumed AssertedContext' relationship must be drawn horizontally relative to its connected elements. Permitted connections of the 'assumed AssertedContext' (inherited from the 'AssertedContext'):</p> <ul style="list-style-type: none"> • From (source element) : 'ArtifactReference' (as a Context) or a 'Claim' element (as a necessary context (i.e. a precondition)) • To (target element) : 'Assertion' type (e.g. 'Claim').

A.5.4 'axiomatic AssertedContext'

TABLE A.23: 'axiomatic AssertedContext' visual representation identity

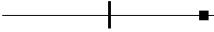
Visual representation identify of		'axiomatic AssertedContext'
1	Visual representation	
2	Semantics	An 'AssertedContext' that is intentionally declared as axiomatically true, so that no further argumentation is needed.
3	Visual variable used	Inherited from the design constraints of an 'AssertedContext': <ul style="list-style-type: none"> • Shape: 'A line' and 'a square'. • Location/Position: The square must be located near one of the line-end that is used as a pointer to the targeted element. The other line-end is used to indicate the source element of the relationship. When used in the diagram the 'AssertedContext' must be drawn horizontally relative to its source and target elements. • Size: '1 pt' line size thickness for the line; '0.30' cm for the width and height of the square (adjustable relative to the application in a diagram). • Texture: 'Solid' for the texture of the line. • Brightness: 'Dark' for the square.
		Inherited from the 'axiomatic' design constraints: <ul style="list-style-type: none"> • Shape: 'A line' - must be located in the middle of the line and positioned vertically if the line of the 'AssertedRelationship' type drawn horizontally, and positioned horizontally if the line of the 'AssertedRelationship' type drawn vertically. • Location/Position: 'In the middle of the line'. • Size: '3pt' as the size of the line. • Texture: 'Solid' for the texture of the line.
4	Visual inheritance info	Inherited design constraints of 'AssertedContext' and 'axiomatic'.
5	Design rationale	
	Avoiding clash with the existing notations	No existing notation in the same domain is used the visual representation of the 'axiomatic AssertedContext'.
	Adopting existing visual notation design principles	
	Semiotic clarity	The visual representation of the 'axiomatic AssertedContext' looks identical with the visual representation of the 'axiomatic AssertedArtifactContext' due to consider graphic complexity of a notation. They can be differentiated by identifying the source and target elements that are connected to them.
	Semantic transparency	Considered as an index visual representation for the user who already learn about the SACM 'AssertedContext' and 'axiomatic'.
	Perceptual discriminability	The visual representation of the 'axiomatic AssertedContext' can be differentiated with other visual representations.
	Redundant coding	The visual representation of the 'axiomatic AssertedContext' is created using multiple visual variables.
	Notation Usability	
	Easy to draw	The visual representation of the 'axiomatic AssertedContext' is considered relatively easy to draw.
	Use of colour	Not using Colour to create the visual representation of the 'axiomatic AssertedContext'.
Tool support implementation	The visual representation of the 'axiomatic AssertedContext' is considered feasible to be implemented in a tool support.	
Other constraints (if any)	Considering graphic complexity (i.e. managing the number of the visual representations of a notation).	

Table A.23 continued from previous page

Visual representation identify of		‘axiomatic AssertedContext’
6	Compositional rules when used in a diagram	<p>An ‘axiomatic AssertedContext’ relationship must be drawn horizontally relative to its connected elements. Permitted connections of the ‘axiomatic AssertedContext’ (inherited from the ‘AssertedContext’):</p> <ul style="list-style-type: none"> • From (source element) : ‘ArtifactReference’ (as a Context) or a ‘Claim’ element (as a necessary context (i.e. a precondition)) • To (target element) : ‘Assertion’ type (e.g. ‘Claim’).

A.5.5 ‘defeated AssertedContext’

TABLE A.24: ‘defeated AssertedContext’ visual representation identity


Visual representation identify of		‘defeated AssertedContext’
1	Visual representation	
2	Semantics	An ‘AssertedContext’ that is defeated by counter argument/evidence.
3	Visual variable used	<p>Inherited from the design constraints of an ‘AssertedContext’:</p> <ul style="list-style-type: none"> • Shape: ‘A line’ and ‘a square’. • Location/Position: The square must be located near one of the line-end that is used as a pointer to the targeted element. The other line-end is used to indicates the source element of the relationship. When used in the diagram the ‘AssertedContext’ must be drawn horizontally relative to its source and target elements. • Size: ‘1 pt’ line size thickness for the line; ‘0.30’ cm for the width and height of the square (adjustable relative to the application in a diagram). • Texture: ‘Solid’ for the texture of the line. • Brightness: ‘Dark’ for the square.
		<p>Inherited from the ‘defeated’ design constraints:</p> <ul style="list-style-type: none"> • Shape: ‘A-double-lines’ that form a cross symbol. • Location: The cross symbol is placed in the middle of the line. • Size: ‘1 pt’ line size thickness for the lines that form a cross symbol; The height of cross must be drawn relatively in the size of the line-head of the relationship. • Texture: ‘Solid’ for the texture of the line.
4	Visual inheritance info	Inherited design constraints of ‘AssertedContext’ and ‘defeated’.
5	Design rationale	
	Avoiding clash with the existing notations	No existing notation in the same domain is used the visual representation of the ‘defeated AssertedContext’.
	Adopting existing visual notation design principles	
	Semiotic clarity	The visual representation of the ‘defeated AssertedContext’ looks identical with the visual representation of the ‘defeated AssertedArtifactContext’ due to consider graphic complexity of a notation. They can be differentiated by identifying the source and target elements that are connected to them.
	Semantic transparency	Considered as an index visual representation for the user who already learn about the SACM ‘AssertedContext’ and ‘defeated’.
Perceptual discriminability	The visual representation of the ‘defeated AssertedContext’ can be differentiated with other visual representations.	
Redundant coding	The visual representation of the ‘defeated AssertedContext’ is created using multiple visual variables.	

Table A.24 continued from previous page

Visual representation identify of		'defeated AssertedContext'
	Notation Usability	
	Easy to draw	The visual representation of the 'defeated AssertedContext' is considered relatively easy to draw.
	Use of colour	Not using Colour to create the visual representation of the 'defeated AssertedContext'.
	Tool support implementation	The visual representation of the 'defeated AssertedContext' is considered feasible to be implemented in a tool support.
	Other constraints (if any)	Considering graphic complexity (i.e. managing the number of the visual representations of a notation).
6	Compositional rules when used in a diagram	<p>A 'defeated AssertedContext' relationship must be drawn horizontally relative to its connected elements. Permitted connections of the 'defeated AssertedContext' (inherited from the 'AssertedContext'):</p> <ul style="list-style-type: none"> • From (source element) : 'ArtifactReference' (as a Context) or a 'Claim' element (as a necessary context (i.e. a precondition)) • To (target element) : 'Assertion' type (e.g. 'Claim').

A.5.6 'asCited AssertedContext'

TABLE A.25: 'asCited AssertedContext' visual representation identity

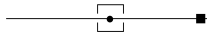
Visual representation identify of		'asCited AssertedContext'
1	Visual representation	
2	Semantics	An 'AssertedContext' which cites another 'AssertedContext'.
3	Visual variable used	<p>Inherited from the design constraints of an 'AssertedContext':</p> <ul style="list-style-type: none"> • Shape: 'A line', 'a square' and 'A Circle'. • Location/Position: The square must be located near one of the line-end that is used as a pointer to the targeted element. The circle must be located in the middle of the line. The other line-end is used to indicate the source element of the relationship. When used in the diagram the 'AssertedContext' must be drawn horizontally relative to its source and target elements. • Size: '1 pt' line size thickness for the line; '0.25' cm for the width and height of the circle; '0.30' cm for the width and height of the square (adjustable relative to the application in a diagram). • Texture: 'Solid' for the texture of the line. • Brightness: 'Dark' for the square and the circle.
		<p>Inherited from the 'asCited' design constraints:</p> <ul style="list-style-type: none"> • Shape: 'Lines' that form square brackets symbol ('[]'). • Location: The 'AssertedContext' must be located within the square brackets. • Size: '1 pt' line size thickness for the lines that form the square brackets. • Texture: 'Solid' for the texture of the line.
4	Visual inheritance info	Inherited design constraints of 'AssertedContext' and 'asCited'.
	Design rationale	
	Avoiding clash with the existing notations	No existing notation in the same domain is used the visual representation of the 'asCited AssertedContext'.
	Adopting existing visual notation design principles	

Table A.25 continued from previous page

Visual representation identify of		'asCited AssertedContext'
	Semiotic clarity	The visual representation of the 'asCited AssertedContext' looks identical with the visual representation of the 'asCited AssertedArtifactContext' due to consider graphic complexity of a notation. They can be differentiated by identifying the source and target elements that are connected to them.
	Semantic transparency	Considered as an index visual representation for the user who already learn about the SACM 'AssertedContext' and 'asCited'.
	Perceptual discriminability	The visual representation of the 'asCited AssertedContext' can be differentiated with other visual representations.
	Redundant coding	The visual representation of the 'asCited AssertedContext' is created using multiple visual variables.
	Notation Usability	
	Easy to draw	The visual representation of the 'asCited AssertedContext' is considered relatively easy to draw.
	Use of colour	Not using Colour to create the visual representation of the 'asCited AssertedContext'.
	Tool support implementation	The visual representation of the 'asCited AssertedContext' is considered feasible to be implemented in a tool support.
	Other constraints (if any)	Considering graphic complexity (i.e. managing the number of the visual representations of a notation).
6	Compositional rules when used in a diagram	An 'asCited AssertedContext' relationship must be drawn horizontally relative to its connected elements. Permitted connections of the 'asCited AssertedContext' (inherited from the 'AssertedContext'): <ul style="list-style-type: none"> • From (source element) : 'ArtifactReference' (as a Context) or a 'Claim' element (as a necessary context (i.e. a precondition)) • To (target element) : 'Assertion' type (e.g. 'Claim').

A.5.7 'abstract AssertedContext'

TABLE A.26: 'abstract AssertedEvidence' visual representation identity

Visual representation identify of		'abstract AssertedContext'
1	Visual representation	-----■
2	Semantics	An 'AssertedContext' that is declared as part of a pattern or template.
3	Visual variable used	Inherited from the design constraints of an 'AssertedContext': <ul style="list-style-type: none"> • Shape: 'A line' and 'a square'. • Location/Position: The square must be located near one of the line-end that is used as a pointer to the targeted element. The other line-end is used to indicates the source element of the relationship. When used in the diagram the 'AssertedContext' must be drawn horizontally relative to its source and target elements. • Size: '1 pt' line size thickness for the line; '0.30' cm for the width and height of the square (adjustable relative to the application in a diagram). • Texture: 'Dash line' - line texture of the Rectangle (respecting the inherited 'isAbstract' design constraint). • Brightness: 'Dark' for the square.
4	Visual inheritance info	Inherited design constraints of 'AssertedContext' and 'isAbstract'.
	Design rationale	

Table A.26 continued from previous page

Visual representation identify of		‘abstract AssertedContext’
	Avoiding clash with the existing notations	No existing notation in the same domain is used the visual representation of the ‘abstract AssertedContext’.
	Adopting existing visual notation design principles	
	Semiotic clarity	The visual representation of the ‘abstract AssertedContext’ looks identical with the visual representation of the ‘abstract AssertedArtifactContext’ due to consider graphic complexity of a notation. They can be differentiated by identifying the source and target elements that are connected to them.
	Semantic transparency	Considered as an index visual representation for the user who already learn about the SACM ‘AssertedContext’ and ‘abstract’.
	Perceptual discriminability	The visual representation of the ‘abstract AssertedContext’ can be differentiated with other created visual representation.
	Redundant coding	The visual representation of the ‘abstract AssertedContext’ is created using multiple visual variables.
	Notation Usability	
	Easy to draw	The visual representation of the ‘abstract AssertedContext’ is considered relatively easy to draw.
	Use of colour	Not using Colour to create the visual representation of the ‘abstract AssertedContext’.
	Tool support implementation	The visual representation of the ‘abstract AssertedContext’ is considered feasible to be implemented in a tool support.
	Other constraints (if any)	Considering graphic complexity (i.e. managing the number of the visual representations of a notation).
6	Compositional rules when used in a diagram	An ‘abstract AssertedContext’ relationship must be drawn horizontally relative to its connected elements. Permitted connections of the ‘abstract AssertedContext’ (inherited from the ‘AssertedContext’): <ul style="list-style-type: none"> • From (source element) : ‘ArtifactReference’ (as a Context) or a ‘Claim’ element (as a necessary context (i.e. a precondition)) • To (target element) : ‘Assertion’ type (e.g. ‘Claim’).

A.5.8 ‘counter AssertedContext’

TABLE A.27: ‘counter AssertedEvidence’ visual representation identity

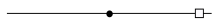
Visual representation identify of		‘counter AssertedContext’
1	Visual representation	
2	Semantics	An ‘AssertedContext’ that counters its declared purposes.

Table A.27 continued from previous page

Visual representation identify of		'counter AssertedContext'
3	Visual variable used	Inherited from the design constraints of an 'AssertedContext': <ul style="list-style-type: none"> • Shape: 'A line', 'a square' and 'A Circle'. • Location/Position: The square must be located near one of the line-end that is used as a pointer to the targeted element. The circle must be located in the middle of the line. The other line-end is used to indicates the source element of the relationship. When used in the diagram the 'AssertedContext' must be drawn horizontally relative to its source and target elements. • Size: '1 pt' line size thickness for the line; '0.25' cm for the width and height of the circle; '0.30' cm for the width and height of the square (adjustable relative to the application in a diagram). • Texture: 'Solid' for the texture of the line. • Brightness: 'Light' for the square (applying the inherited design constraint of the 'isCounter'); and 'dark' for the circle.
4	Visual inheritance info	Inherited design constraints of 'AssertedContext' and 'isCounter'.
5	Design rationale	
	Avoiding clash with the existing notations	No existing notation in the same domain is used the visual representation of the 'counter AssertedContext'.
	Adopting existing visual notation design principles	
	Semiotic clarity	The visual representation of the 'counter AssertedContext' looks identical with the visual representation of the 'counter AssertedArtifactContext' due to consider graphic complexity of a notation. They can be differentiated by identifying the source and target elements that are connected to them.
	Semantic transparency	Considered as an index visual representation for the user who already learn about the SACM 'AssertedContext' and 'isCounter'.
	Perceptual discriminability	The visual representation of the 'counter AssertedContext' can be differentiated with other created visual representation.
	Redundant coding	The visual representation of the 'counter AssertedContext' is created using multiple visual variables.
	Notation Usability	
	Easy to draw	The visual representation of the 'counter AssertedContext' is considered relatively easy to draw.
Use of colour	Not using Colour to create the visual representation of the 'counter AssertedContext'.	
Tool support implementation	The visual representation of the 'counter AssertedContext' is considered feasible to be implemented in a tool support.	
Other constraints (if any)	Considering graphic complexity (i.e. managing the number of the visual representations of a notation).	
6	Compositional rules when used in a diagram	A 'counter AssertedContext' relationship must be drawn horizontally relative to its connected elements. Permitted connections of the 'counter AssertedContext' (inherited from the 'AssertedContext'): <ul style="list-style-type: none"> • From (source element) : 'ArtifactReference' (as a Context) or a 'Claim' element (as a necessary context (i.e. a precondition)) • To (target element) : 'Assertion' type (e.g. 'Claim').

A.6 ‘AssertedArtifactSupport’ and its variants

Table A.28 presents the visual representation identity of ‘AssertedArtifactSupport’. The other types of ‘AssertedArtifactSupport’ element are created based on visual inheritance combining the design constraints of the ‘AssertedArtifactSupport’ and the design constraints of a particular inherited attribute. For example, ‘assumed AssertedArtifactSupport’ element is created by combining design constraints of the ‘AssertedArtifactSupport’ and ‘assumed’. The illustration of the visual representation identity of the other types of ‘AssertedArtifactSupport’ element is similar with the visual representation identity of the other types of ‘AssertedInference’ element.

A.6.1 ‘AssertedArtifactSupport’

TABLE A.28: ‘AssertedArtifactSupport’ visual representation identity

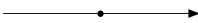
Visual representation identify of		‘AssertedArtifactSupport’
1	Visual representation	
2	Semantics	The truth of the assertions associated with an artifact are supported by the assertions that are associated with one or more other artifacts. Note: this can be an ambiguous relationship if the nature of these ‘Assertions’ is unclear. In such cases, it would be clearer to declare explicit ‘AssertedInferences’ between ‘Claims’ drawn out from the ‘ArtifactReference’.
3	Visual variable used	<ul style="list-style-type: none"> • Shape: ‘Arrowhead-line’ and ‘Circle’. • Location: The edge without an arrow must be connected to the source element of the relationship that must be an ‘ArtifactReference’ (evidence) element. The edge with an arrow must be connected to the target element of the relationship that must be the ‘ArtifactReference’ element. The source element must be located vertically (below) relative to the targeted element. • Size: ‘1 pt’ line size thickness for the line; ‘0.25’ cm for the width and height of the circle (adjustable relative to the application in a diagram). • Texture: ‘Solid’ for the texture of the line. • Brightness: ‘Dark’ for the arrowhead and the circle.
4	Visual inheritance info	Inherited design constraints from (abstract) ‘AssertedRelationship’ class: must be drawn as a line with a 2D-type line-head as a pointer to indicates the target element of the relationship; and the other line-end (without line-head) must be used as a pointer to indicates the source element of the relationship.
5	Design rationale	
	Avoiding clash with the existing notations	No existing notation in the same domain is used the visual representation of the ‘AssertedArtifactSupport’.
	Adopting existing visual notation design principles	
	Semiotic clarity	The visual representation of the ‘AssertedArtifactSupport’ looks identical with the visual representation of the ‘AssertedInference’ and ‘AssertedEvidence’ due to consider graphic complexity of a notation. They can be differentiated by identifying the source and target elements that are connected to them.

Table A.28 continued from previous page

Visual representation identify of		‘AssertedArtifactSupport’
	Semantic transparency	The ‘AssertedArtifactSupport’ relationship is considered as a symbol-type visual representation. The notation user needs to learn in order to understand the meaning of it.
	Perceptual discriminability	The visual representation of the ‘AssertedArtifactSupport’ can be differentiated with other visual representations.
	Redundant coding	The visual representation of the ‘AssertedArtifactSupport’ is created using multiple visual variables.
	Notation Usability	
	Easy to draw	The visual representation of the ‘AssertedArtifactSupport’ is considered relatively easy to draw.
	Use of colour	Not using Colour to create the visual representation of the ‘AssertedArtifactSupport’.
	Tool support implementation	A dot that is placed in the middle of the line is added due to consider the tool support implementation factor (input from notation stakeholder). Therefore, the visual representation of the ‘AssertedArtifactSupport’ is considered feasible to be implemented in a tool support.
	Other constraints (if any)	Considering graphic complexity (i.e. managing the number of the visual representations of a notation).
6	Compositional rules when used in a diagram	An ‘AssertedArtifactSupport’ relationship must be drawn vertically relative to its connected elements. Permitted connection of the ‘AssertedArtifactSupport’ can be described as follow: <ul style="list-style-type: none"> • The source element must be: ‘ArtifactReference’ (as evidence) and located vertically (below) relative to the targeted element. • The target element must be: ‘ArtifactReference’ (can be as a reference to a contextual information or a reference to another evidential information).

A.7 ‘AssertedArtifactContext’ and its variants

Table A.29 presents the visual representation identity of ‘AssertedArtifactContext’. The other types of ‘AssertedArtifactContext’ element are created based on visual inheritance combining the design constraints of the ‘AssertedArtifactContext’ and the design constraints of a particular inherited attribute. For example, ‘axiomatic AssertedArtifactContext’ element is created by combining design constraints of the ‘AssertedArtifactSupport’ and ‘axiomatic’. The illustration of the visual representation identity of the other types of ‘AssertedArtifactContext’ element is similar with the visual representation identity of the other types of ‘AssertedContext’ element.

A.7.1 ‘AssertedArtifactContext’

TABLE A.29: ‘AssertedArtifactContext’ visual representation identity

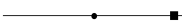
Visual representation identify of		‘AssertedArtifactContext’
1	Visual representation	

Table A.29 continued from previous page

Visual representation identify of		'AssertedArtifactContext'
2	Semantics	One or more other artifacts provide the necessary context in which the assertions associated with another artifact should be understood. Note: this can be an ambiguous relationship if the nature of these 'Assertions' is unclear. In such cases, it would be clearer to declare explicit 'AssertedContext' between 'Claim's drawn out from the 'ArtifactReference'.
3	Visual variable used	<ul style="list-style-type: none"> • Shape: 'A line', 'a square' and 'A Circle'. • Location/Position: The square must be located near one of the line-end that is used as a pointer to the targeted element. The circle must be located in the middle of the line. The other line-end is used to indicates the source element of the relationship. When used in the diagram the 'AssertedArtifactContext' must be drawn horizontally relative to its source and target elements. • Size: '1 pt' line size thickness for the line; '0.25' cm for the width and height of the circle; '0.30' cm for the width and height of the square (adjustable relative to the application in a diagram). • Texture: 'Solid' for the texture of the line. • Brightness: 'Dark' for the square and the circle.
4	Visual inheritance info	Inherited design constraints from (abstract) 'AssertedRelationship' class: must be drawn as a line with a 2D-type line-head as a pointer to indicates the target element of the relationship; and the other line-end (without line-head) must be used as a pointer to indicates the source element of the relationship.
5	Design rationale	
	Avoiding clash with the existing notations	No existing notation in the same domain is used the visual representation of the 'AssertedArtifactContext'.
	Adopting existing visual notation design principles	
	Semiotic clarity	The visual representation of the 'AssertedArtifactContext' looks identical with the visual representation of the 'AssertedContext' due to consider graphic complexity of a notation. They can be differentiated by identifying the source and target elements that are connected to them.
	Semantic transparency	The 'AssertedArtifactContext' relationship is considered as a symbol-type visual representation. The user needs to learn in order to understand the meaning of it.
	Perceptual discriminability	The visual representation of the 'AssertedArtifactContext' can be differentiated with other visual representations.
	Redundant coding	The visual representation of the 'AssertedArtifactContext' is created using multiple visual variables.
	Notation Usability	
	Easy to draw	The visual representation of the 'AssertedArtifactContext' is considered relatively easy to draw.
	Use of colour	Not using Colour to create the visual representation of the 'AssertedArtifactContext'.
Tool support implementation	A dot that is placed in the middle of the line is added due to consider the tool support implementation factor (input from notation stakeholder). Therefore, the visual representation of the 'AssertedArtifactContext' is considered feasible to be implemented in a tool support.	
Other constraints (if any)	Considering graphic complexity (i.e. managing the number of the visual representations of a notation).	

Table A.29 continued from previous page

Visual representation identify of		‘AssertedArtifactContext’
6	Compositional rules when used in a diagram	<p>An ‘AssertedArtifactContext’ relationship must be drawn horizontally relative to its connected elements. Permitted connections of the ‘AssertedArtifactContext’:</p> <ul style="list-style-type: none"> • From (source element) : ‘ArtifactReference’ as a reference to a contextual information. • To (target element) : ‘ArtifactReference’ as a reference to a evidential information.

A.8 ‘ArgumentReasoning’

TABLE A.30: ‘ArgumentReasoning’ visual representation identity

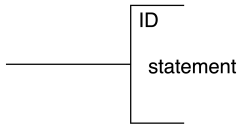
Visual representation identify of		‘ArgumentReasoning’
1	Visual representation	
2	Semantics	The ‘AssertedRelationship’ that relates one or more ‘Claims’ (premises) to another ‘Claim’ (conclusion), or evidence cited by an ‘ArtifactReference’ to a ‘Claim’, may not always be obvious. In such cases ‘ArgumentReasoning’ can be used to provide further description of the reasoning involved.
3	Visual variable used	<ul style="list-style-type: none"> • Shape: ‘Lines’ combined together forming an annotation symbol. • Size: ‘1 point’ for the size of the line. • Position: Horizontally (left or right side) relative to the targeted element.
		<p>Inherited from the ‘gid’ attribute design constraint:</p> <ul style="list-style-type: none"> • Location: ‘Top-Left’ within the element.
4	Visual inheritance info	Inherited design constraint of ‘gid’ attribute.
5	Design rationale	
	Avoiding clash with the existing notations	The visual representation of the ‘ArgumentReasoning’ is inspired from the Flowchart annotation symbol and the BPMN text annotation symbol. The adoption of the symbol due consider to reduce learning process at least for the notation user who familiar with the existing (Flowchart annotation and the BPMN text annotation) symbol.
	Adopting existing visual notation design principles	
	Semiotic clarity	The annotation symbol in SACM notation is only used to represent the ‘ArgumentReasoning’ element.
	Semantic transparency	Considered as a symbol-type visual representation for the notation user who have no prior experience related to used of the Flowchart annotation or the BPMN text annotation symbol because they need to learn in order to understand the meaning of it.
Perceptual discriminability	The visual representation of the ‘ArgumentReasoning’ can be differentiated with other visual representations.	
Redundant coding	The visual representation of the ‘ArgumentReasoning’ is created using multiple visual variables.	

Table A.30 continued from previous page

Visual representation identify of		‘ArgumentReasoning’
	Notation Usability	
	Easy to draw	The visual representation of the ‘ArgumentReasoning’ is considered relatively easy to draw.
	Use of colour	Not using Colour to create the visual representation of the ‘ArgumentReasoning’.
	Tool support implementation	The visual representation of the ‘ArgumentReasoning’ is considered feasible to be implemented in a tool support.
	Other constraints (if any)	N/A.
6	Compositional rules when used in a diagram	An ‘ArgumentReasoning’ can be connected to ‘AssertedInference’ or ‘AssertedEvidence’ relationship in order to provides additional description or explanation of the asserted relationship. When used in a diagram, ‘ArgumentReasoning’ must be located on the left or right of the targeted element.

A.9 ‘metaClaim’

TABLE A.31: ‘metaClaim’ visual representation identity


Visual representation identify of		‘metaClaim’
1	Visual representation	
2	Semantics	‘metaClaim’:‘Claim’[0..*] - references Claims concerning (i.e., about) the ‘Assertion’ (e.g., regarding the confidence in the ‘Assertion’)
3	Visual variable used	<ul style="list-style-type: none"> • Shape: ‘A crow’s foot line’. • Texture: ‘Solid’ for the texture of the line. • Size: ‘1 point’ for the size of the line thickness. • Position/Location: ‘Horizontal/Vertical’ relative to its source and target elements when used in a diagram.
4	Visual inheritance info	N/A.
5	Design rationale	
	Avoiding clash with the existing notations	No existing notation in the same domain is used the visual representation of the ‘metaClaim’.
	Adopting existing visual notation design principles	
	Semiotic clarity	The crow’s foot line is created and used only for representing the semantics of ‘metaClaim’.
	Semantic transparency	Considered as a symbol-type visual representation because the notation user needs to learn in order to understand the meaning of it.
	Perceptual discriminability	The visual representation of the ‘metaClaim’ can be differentiated with other visual representations.
	Redundant coding	The visual representation of the ‘metaClaim’ is created using multiple visual variables.
	Notation Usability	
	Easy to draw	The visual representation of the ‘metaClaim’ is considered relatively easy to draw.
	Use of colour	Not using Colour to create the visual representation of the ‘metaClaim’.
Tool support implementation	The visual representation of the ‘metaClaim’ is considered feasible to be implemented in a tool support.	
Other constraints (if any)		N/A.

Table A.31 continued from previous page

Visual representation identify of		‘metaClaim’
6	Compositional rules when used in a diagram	When used in a diagram, a ‘metaClaim’ relationship starts from its source element which is must be a ‘Claim’ element and pointing to its target element that must be a type of an ‘Assertion’ such as ‘Claim’ element. The ‘Claim’ as the source element of the ‘metaClaim’ relationship can be used as a ‘Claim’ that arguing about the ‘Assertion’ that the ‘metaClaim’ relationship is pointing. In this case, the crow’s foot is important to indicates the element that is being argued by the ‘Claim’ of the ‘metaClaim’ relationship.

A.10 ‘ArgumentPacakge’ and its variants

A.10.1 ‘ArgumentPacakge’

TABLE A.32: ‘ArgumentPacakge’ visual representation identity

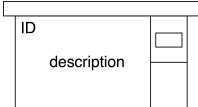
Visual representation identify of		‘ArgumentPacakge’
1	Visual representation	
2	Semantics	‘ArgumentPacakges’ contain structured arguments. These arguments are composed of ‘ArgumentAssets’. ‘ArgumentPacakges’ elements can also be nested.
3	Visual variable used	<ul style="list-style-type: none"> • Shape: ‘Rectangles’ that combined together to form a package icon. • Brightness: ‘Light’ as the background within the icon. • Size: ‘1 point’ as the line size thickness to form the icon. • Position/Location: The position of the ‘ArgumentPacakge’ in diagram can be ‘Vertical/Horizontal’ relative to its connected elements; A rectangle placed within the top-right compartment of the icon. • Texture: ‘Solid’ as the texture of the line that used to form the icon.
		Inherited from the ‘gid’ attribute design constraint: <ul style="list-style-type: none"> • Location: ‘Top-Left’ within the element.
4	Visual inheritance info	Inherited design constraint of ‘gid’ attribute.
5	Design rationale	
	Avoiding clash with the existing notations	No existing notation in the same domain is used the visual representation of the ‘ArgumentPacakge’.
	Adopting existing visual notation design principles	
	Semiotic clarity	The icon visual representation of the ‘ArgumentPacakge’ is only used for representing the ‘ArgumentPacakge’ element.
	Semantic transparency	Considered as an icon-type visual representation. The notation user might understand the meaning of it when looking at the visual representation.
	Perceptual discriminability	The visual representation of the ‘ArgumentPacakge’ can be differentiated with other visual representations.
	Redundant coding	The visual representation of the ‘ArgumentPacakge’ is created using multiple visual variables.
Notation Usability		
Easy to draw	The visual representation of the ‘ArgumentPacakge’ is considered relatively easy to draw.	

Table A.32 continued from previous page

Visual representation identify of		'ArgumentPacakge'
	Use of colour	Not using Colour to create the visual representation of the 'ArgumentPacakge'.
	Tool support implementation	The visual representation of the 'ArgumentPacakge' is considered feasible to be implemented in a tool support.
	Other constraints (if any)	N/A.
6	Compositional rules when used in a diagram	When used in a diagram, an 'ArgumentPackage' can be associated to 'ArgumentPackagesInterface' via the 'implements' relationship. An 'ArgumentPackage' can be located vertically/horizontally relative to its connected elements.

A.10.2 'ArgumentPackageInterface'

TABLE A.33: 'ArgumentPackageInterface' visual representation identity

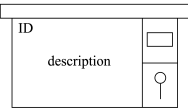
Visual representation identify of		'ArgumentPackageInterface'
1	Visual representation	
2	Semantics	'ArgumentPackageInterfaces' can be used to declare (by means of containing 'ArgumentElement' based citations) the 'ArgumentAssets' contained in an 'ArgumentPackage' that form part of the explicit, declared, interface of the 'ArgumentPackage'.
3	Visual variable used	<p>Inherited from the design constraints of an 'ArgumentPackage':</p> <ul style="list-style-type: none"> • Shape: 'Rectangles' that combined together to form a package icon. • Brightness: 'Light' as the background within the icon. • Size: '1 point' as the line size thickness to form the icon. • Position/Location: The position of the 'ArgumentPackage' in diagram can be 'Vertical/Horizontal' relative to its connected elements; A rectangle placed within the top-right compartment of the icon. • Texture: 'Solid' as the texture of the line that used to form the icon. <p>The following visual variables are used to create the unique visual representation of the 'ArgumentPackageInterface':</p> <ul style="list-style-type: none"> • Shape: 'A circle' and 'a Line' that combined together as illustrated in the figure of point 1 (bottom-right compartment). • Brightness: 'Light' as the background of the circle. • Size: '1 point' as the line size thickness to form the circle and the line. • Position/Location: A circle placed at the upper-part of the line that is drawn vertically. • Texture: 'Solid' as the texture of the line that used to form the circle and the line.
4	Visual inheritance info	Inherited design constraint of 'ArgumentPackage' and 'gid' attribute.
5	Design rationale	
	Avoiding clash with the existing notations	No existing notation in the same domain is used the visual representation of the 'ArgumentPackageInterface'.
	Adopting existing visual notation design principles	
	Semiotic clarity	The icon visual representation of the 'ArgumentPackageInterface' is only used for representing the 'ArgumentPackageInterface' element.

Table A.33 continued from previous page

Visual representation identify of		‘ArgumentPackageInterface’
	Semantic transparency	Considered as an icon-type visual representation. The notation user might understand the meaning of it when looking at the visual representation.
	Perceptual discriminability	The visual representation of the ‘ArgumentPackageInterface’ can be differentiated with other visual representations.
	Redundant coding	The visual representation of the ‘ArgumentPackageInterface’ is created using multiple visual variables.
	Notation Usability	
	Easy to draw	The visual representation of the ‘ArgumentPackageInterface’ is considered relatively easy to draw.
	Use of colour	Not using Colour to create the visual representation of the ‘ArgumentPackageInterface’.
	Tool support implementation	The visual representation of the ‘ArgumentPackageInterface’ is considered feasible to be implemented in a tool support.
	Other constraints (if any)	N/A.
6	Compositional rules when used in a diagram	<ul style="list-style-type: none"> • The elements within the ‘ArgumentPackageInterface’ must be a type of citation element, e.g., ‘asCited Claim’, where the cited element is the ‘ArgumentAssets’ or argumentation elements that is located within an ‘ArgumentPackage’. • An ‘implements’ relationship can be used as a relationship to associate between ‘ArgumentPackage’ and ‘ArgumentPackageInterface’.

A.10.3 ‘implements’

TABLE A.34: ‘implements’ visual representation identity

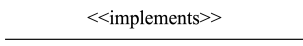
Visual representation identify of		‘implements’
1	Visual representation	
2	Semantics	‘implements’:‘ArgumentPackage’[1] – a reference to the ‘ArgumentPackage’ which the ‘ArgumentPackageInterface’ declares.
3	Visual variable used	<ul style="list-style-type: none"> • Shape: ‘A line’. • Texture: ‘Solid’ for the texture of the line. • Size: ‘1 point’ for the size of the line thickness. • Position: ‘Horizontal/Vertical’ relative to its connected elements when used in a diagram. A keyword ‘«implements»’ attached to the line that can be placed around the middle of it.
4	Visual inheritance info	N/A.
	Design rationale	
	Avoiding clash with the existing notations	No existing notation in the same domain is used the visual representation of the ‘implements’ relationship.
	Adopting existing visual notation design principles	
5	Semiotic clarity	A line with an ‘implements’ keyword attached to it is created and used only for representing the semantics of ‘implements’.
	Semantic transparency	Considered as a symbol-type visual representation because the notation user needs to learn in order to understand the meaning of it.
	Perceptual discriminability	The visual representation of the ‘implements’ can be differentiated with other visual representations.

Table A.34 continued from previous page

Visual representation identify of		'implements'
	Redundant coding	The visual representation of the 'implements' is created using multiple visual variables.
	Notation Usability	
	Easy to draw	The visual representation of the 'implements' is considered relatively easy to draw.
	Use of colour	Not using Colour to create the visual representation of the 'implements'.
	Tool support implementation	The visual representation of the 'implements' is considered feasible to be implemented in a tool support.
	Other constraints (if any)	Considering graphic complexity (i.e. managing the number of the visual representations of a notation). The visual representation of the 'implements' is designed to be similar with 'participantPackage' relationship visual representation. They can be differentiated by identifying the keyword that attach to the line and the elements that are connected to the relationship.
6	Compositional rules when used in a diagram	When used in a diagram, an 'implements' relationship must be connected to an 'ArgumentPacakge' (at one line-end) and an 'ArgumentPacakgeInterface' (at the other line-end). An 'implements' relationship can be drawn horizontally or vertically relative to its connected elements.

A.10.4 'ArgumentPackageBinding'

TABLE A.35: 'ArgumentPackageBinding' visual representation identity

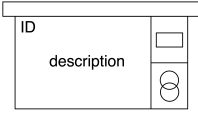
Visual representation identify of		'ArgumentPackageBinding'
1	Visual representation	
2	Semantics	'ArgumentPackageBindings' can be used to map resolved dependencies between the 'Claims' of two or more 'ArgumentPackages'. For example, one 'ArgumentPackage' may contain a 'Claim' that 'needsSupport' (i.e. currently has no supporting argument). An 'ArgumentPackage-Binding' can be used to record the mapping by means of containing a structured argument linking 'ArgumentElements' that cite the 'Claims' in question. 'ArgumentPackageBinding' is a sub type of 'ArgumentPackage', it is used to record the argument that connects the arguments of two or more 'ArgumentPackages'
3	Visual variable used	<p>Inherited from the design constraints of an 'ArgumentPackage':</p> <ul style="list-style-type: none"> • Shape: 'Rectangles' that combined together to form a package icon. • Brightness: 'Light' as the background within the icon. • Size: '1 point' as the line size thickness to form the icon. • Position/Location: The position of the 'ArgumentPackage' in diagram can be 'Vertical/Horizontal' relative to its connected elements; A rectangle placed within the top-right compartment of the icon. • Texture: 'Solid' as the texture of the line that used to form the icon.

Table A.35 continued from previous page

Visual representation identify of		‘ArgumentPackageBinding’
		<p>The following visual variables are used to create the unique visual representation of the ‘ArgumentPackageBinding’:</p> <ul style="list-style-type: none"> • Shape: ‘2 circles’ that are drawn overlap each other as illustrated in figure of point 1 (bottom-right compartment). • Brightness: ‘Light’ as the background of the circles. • Size: ‘1 point’ as the line size thickness to form the circles. • Position/Location: 2 circles placed within the bottom-right of the icon. • Texture: ‘Solid’ as the texture of the line that used to form the circles.
4	Visual inheritance info	Inherited design constraint of ‘ArgumentPackage’ and ‘gid’ attribute.
5	Design rationale	
	Avoiding clash with the existing notations	No existing notation in the same domain is used the visual representation of the ‘ArgumentPackageBinding’.
	Adopting existing visual notation design principles	
	Semiotic clarity	The icon visual representation of the ‘ArgumentPackageBinding’ is only used for representing the ‘ArgumentPackageBinding’ element.
	Semantic transparency	Considered as an icon-type visual representation. The notation user might understand the meaning of it when looking at the visual representation.
	Perceptual discriminability	The visual representation of the ‘ArgumentPackageBinding’ can be differentiated with other visual representations.
	Redundant coding	The visual representation of the ‘ArgumentPackageBinding’ is created using multiple visual variables.
	Notation Usability	
	Easy to draw	The visual representation of the ‘ArgumentPackageBinding’ is considered relatively easy to draw.
	Use of colour	Not using Colour to create the visual representation of the ‘ArgumentPackageBinding’.
Tool support implementation	The visual representation of the ‘ArgumentPackageBinding’ is considered feasible to be implemented in a tool support.	
	Other constraints (if any)	N/A.
6	Compositional rules when used in a diagram	<ul style="list-style-type: none"> • An ‘ArgumentPackageBinding’ element can be connected to, at least, two or more ‘ArgumentPacakgeInterfaces’ to map the ‘ArgumentPackages’. • The ‘participantPackage’ of the ‘ArgumentPackageBinding’ should be only the ‘ArgumentPackages’. • The ‘ArgumentElements’ contained by an ‘ArgumentPackageBinding’ must be ‘ArgumentElement’ citations to ‘ArgumentElements’ contained within the ‘ArgumentPackages’ associated by the ‘participantPackage’ association.

A.10.5 ‘participantPackage’

TABLE A.36: ‘participantPackage’ visual representation identity

Visual representation identify of		‘participantPackage’
1	Visual representation	<p style="text-align: center;"><<participantPackage>></p>

Table A.36 continued from previous page

Visual representation identify of		‘participantPackage’
2	Semantics	”‘participantPackage’:‘ArgumentPackageInterface’[2..*] - the ‘ArgumentPackages’ being mapped together by the ‘ArgumentPackageBinding’.
3	Visual variable used	<ul style="list-style-type: none"> • Shape: ‘A line’. • Texture: ‘Solid’ for the texture of the line. • Size: ‘1 point’ for the size of the line thickness. • Position/Location: ‘Horizontal/Vertical’ relative to its connected elements when used in a diagram. A keyword ‘«participantPackage»’ attached to the line that can be placed around the middle of it.
4	Visual inheritance info	N/A.
5	Design rationale	
	Avoiding clash with the existing notations	No existing notation in the same domain is used the visual representation of the ‘participantPackage’ relationship.
	Adopting existing visual notation design principles	
	Semiotic clarity	A line with an ‘participantPackage’ keyword attached to it is created and used only for representing the semantics of ‘participantPackage’.
	Semantic transparency	Considered as a symbol-type visual representation because the notation user needs to learn in order to understand the meaning of it.
	Perceptual discriminability	The visual representation of the ‘participantPackage’ can be differentiated with other visual representations.
	Redundant coding	The visual representation of the ‘participantPackage’ is created using multiple visual variables.
	Notation Usability	
	Easy to draw	The visual representation of the ‘participantPackage’ is considered relatively easy to draw.
	Use of colour	Not using Colour to create the visual representation of the ‘participantPackage’.
Tool support implementation	The visual representation of the ‘participantPackage’ is considered feasible to be implemented in a tool support.	
Other constraints (if any)	Considering graphic complexity (i.e. managing the number of the visual representations of a notation). The visual representation of the ‘participantPackage’ is designed to be similar with ‘implements’ relationship visual representation. They can be differentiated by identifying the keyword that attach to the line and the elements that are connected to the relationship.	
6	Compositional rules when used in a diagram	When used in a diagram, a ‘participantPackage’ relationship must be connected to an ‘ArgumentPacakgeBinding’ (at one line-end) and an ‘ArgumentPacakgeInterface’ (at the other line-end). A ‘participantPackage’ relationship can be drawn horizontally or vertically relative to its connected elements.

A.11 ‘ArgumentGroup’

TABLE A.37: ‘ArgumentGroup’ visual representation identity

Visual representation identify of		‘ArgumentGroup’
1	Visual representation	

Table A.37 continued from previous page

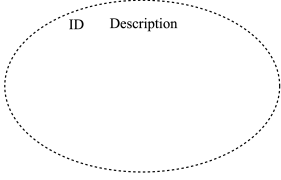
Visual representation identify of		'ArgumentGroup'
		
2	Semantics	'ArgumentGroup' can be used to associate a number of 'ArgumentElements' to a common group (e.g. representing a common type or purpose, or being of interest to a particular stakeholder). The name and the description of the 'ArgumentGroup' should provide the semantic for understanding the 'ArgumentGroup'. 'ArgumentGroups' serve no structural purpose in the formation of the argument network, nor are they meant as a structural packaging mechanism (this should be done using 'ArgumentPackages').
3	Visual variable used	<ul style="list-style-type: none"> • Shape: 'A line' (forming a 2D area). • Location: No specific restriction in drawing the visual representation of the 'ArgumentGroup'; the elements that are considered to be as the same group must be located within the dotted-line. • Size: '1 pt' as the size of the line. • Texture: 'Dotted-line' as the texture of the line. • Brightness: 'Light' as the background of the 'ArgumentGroup'.
		Inherited from the 'gid' attribute design constraint: <ul style="list-style-type: none"> • Location: 'Top-Left' within the element.
4	Visual inheritance info	Inherited design constraint of 'gid' attribute.
5	Design rationale	
	Avoiding clash with the existing notations	No existing notation in the same domain is used the visual representation of the 'ArgumentGroup'.
	Adopting existing visual notation design principles	
	Semiotic clarity	The visual representation of the 'ArgumentGroup' is created only for the representing the semantics of 'ArgumentGroup'.
	Semantic transparency	The visual representation of the 'ArgumentGroup' is considered as a symbol-type visual representation. The notation user need to learn in order to understand the meaning of it.
	Perceptual discriminability	The visual representation of the 'ArgumentGroup' can be differentiated with other visual representations.
	Redundant coding	The visual representation of the 'ArgumentGroup' is created using multiple visual variables.
	Notation Usability	
	Easy to draw	The visual representation of the 'ArgumentGroup' is considered relatively easy to draw.
	Use of colour	Not using Colour to create the visual representation of the 'ArgumentGroup'.
Tool support implementation	The visual representation of the 'ArgumentGroup' is considered feasible to be implemented in a tool support.	
Other constraints (if any)		N/A.

Table A.37 continued from previous page

6	Visual representation identify of in a diagram	'ArgumentGroup'
		When used in a diagram, there is no specific restriction in drawing the visual representation of the 'ArgumentGroup' other than the type of the line in drawing the 'ArgumentGroup' (i.e. using dotted lines). There is also no specific geometrical shape form is defined in creating the 'ArgumentGroup' visual representation. The elements that are considered to be as the same group must be located within the dotted-line.

Appendix B

Empirical study data: SACM notation - GSN

B.1 Assurance case diagram based on case study

Figure B.1 shows the assurance case model presented using SACM notation and Figure B.2 presented using GSN based on the given case study.

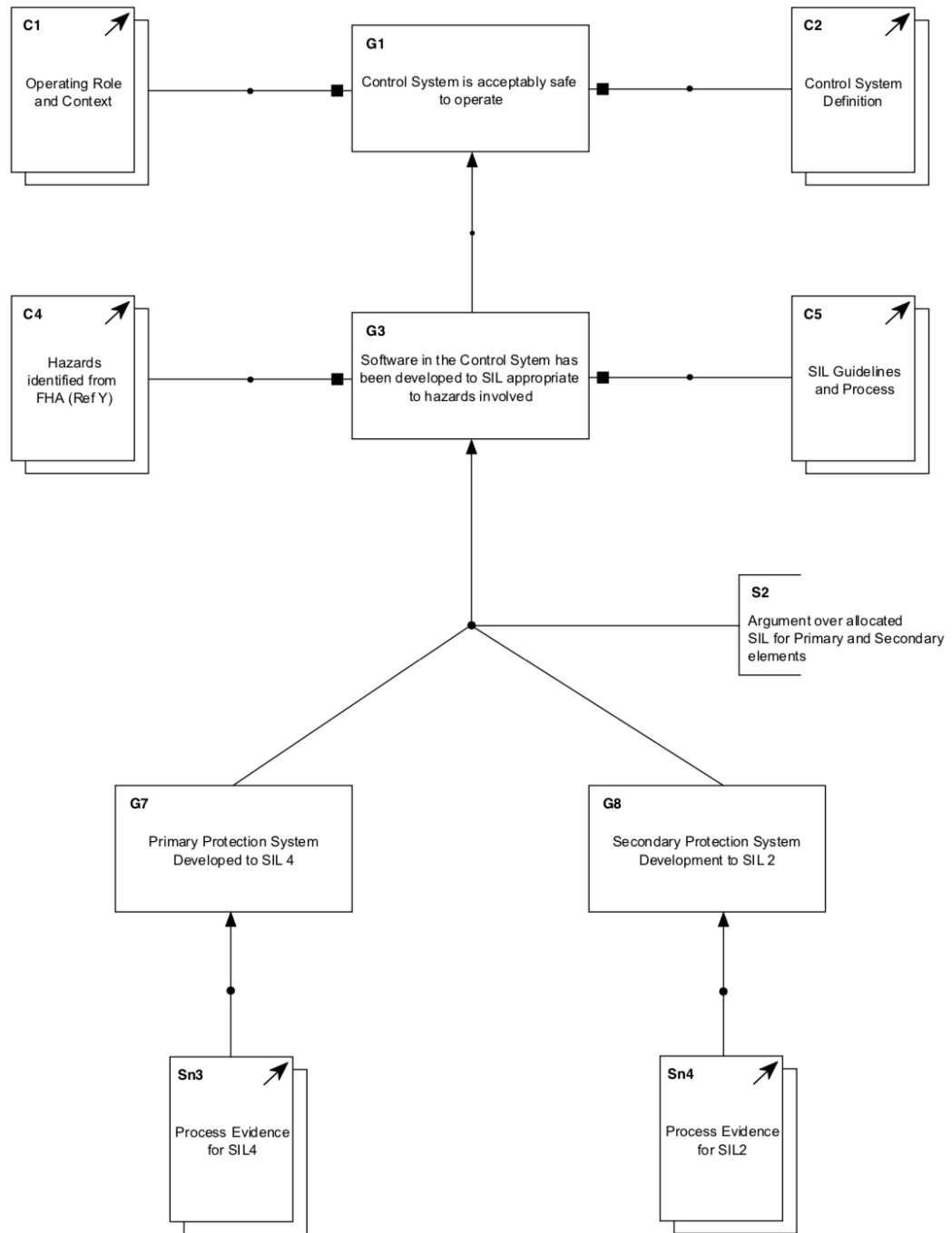


FIGURE B.1: Assurance case diagram constructed using SACM notation based on case study

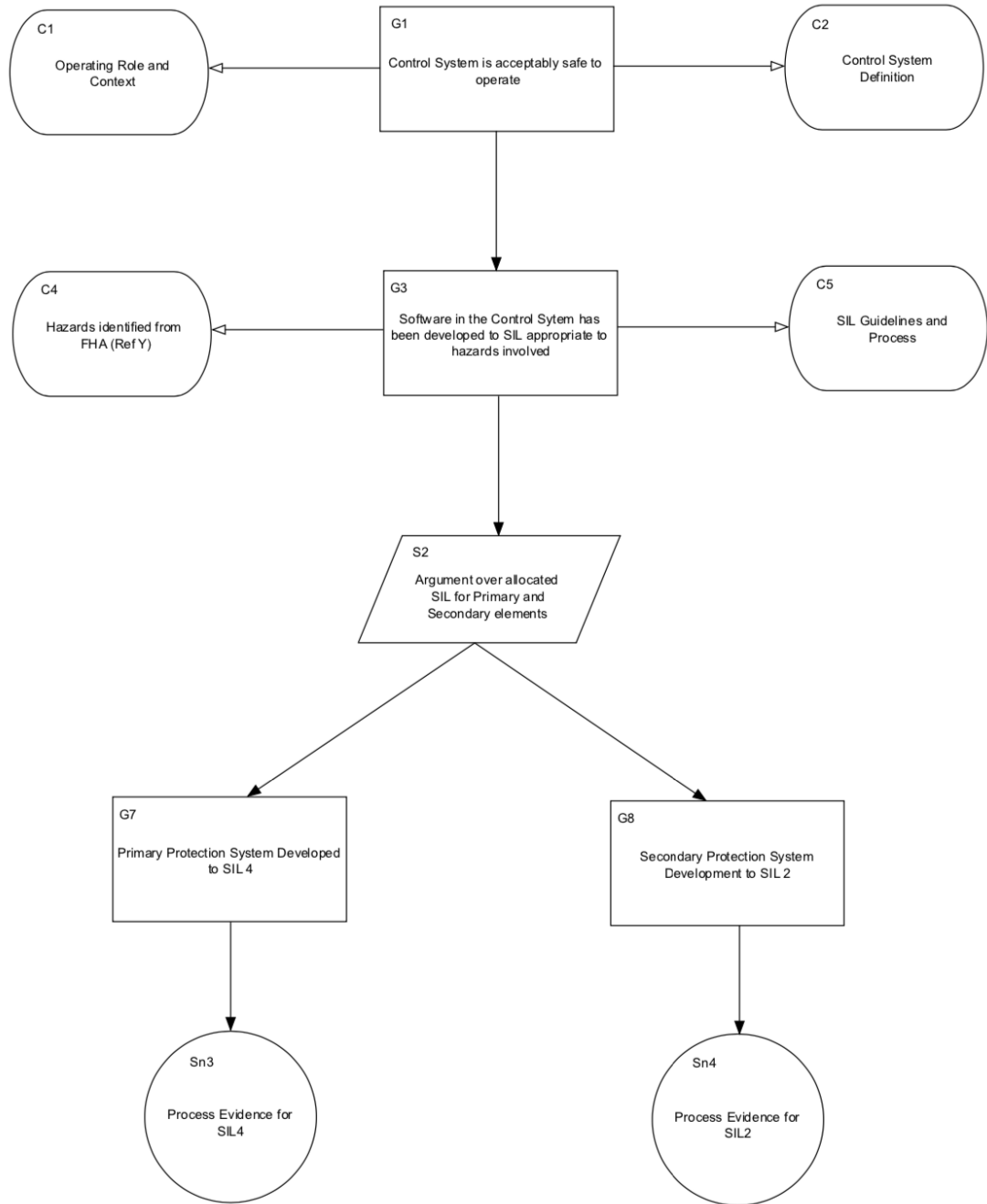


FIGURE B.2: Assurance case diagram constructed using GSN based on case study

B.2 Scoring and timing result

Table B.1 presents Group A participants' scoring and timing result in creating an assurance case model using both SACM notation and GSN, and Table B.2 for Group B participants.

TABLE B.1: Scoring and timing result Group A: SACM notation and GSN

Group A					
SACM notation			GSN		
Participant ID	Score	Timing	Participant ID	Score	Timing
C-SACM-001	100	00:05:34	C-GSN-001	100	00:06:12
C-SACM-002	100	00:07:33	C-GSN-002	83.33	00:06:08
C-SACM-003	80	00:07:38	C-GSN-003	80.95	00:06:27
C-SACM-004	27.5	00:07:44	C-GSN-004	100	00:06:33
C-SACM-005	100	00:10:34	C-GSN-005	100	00:06:49
C-SACM-006	15	00:15:20	C-GSN-006	40.48	00:12:11
C-SACM-007	5	00:15:20	C-GSN-007	80.95	00:12:00
C-SACM-008	77.5	00:15:19	C-GSN-008	88.10	00:12:15
C-SACM-009	57.5	00:14:34	C-GSN-009	23.81	00:08:32
C-SACM-010	45	00:14:05	C-GSN-010	100	00:08:32
C-SACM-011	45	00:13:00	C-GSN-011	23.81	00:08:32
C-SACM-012	60	00:13:00	C-GSN-012	100	00:08:35
C-SACM-013	60	00:13:00	C-GSN-013	78.57	00:09:02
C-SACM-014	80	00:13:00	C-GSN-014	47.62	00:09:20
C-SACM-015	67.5	00:13:00	C-GSN-015	66.67	00:09:15
C-SACM-016	65	00:13:00	C-GSN-016	100	00:09:10
C-SACM-017	27.5	00:13:00	C-GSN-017	100	00:09:50
C-SACM-018	60	00:13:00	C-GSN-018	100	00:09:50
C-SACM-019	47.5	00:13:00	C-GSN-019	23.81	00:09:51
C-SACM-020	47.5	00:13:00	C-GSN-020	42.86	00:09:55
C-SACM-021	62.5	00:12:16	C-GSN-021	100	00:10:21
C-SACM-022	60	00:12:16	C-GSN-022	90.48	00:10:52
C-SACM-023	35	00:12:16	C-GSN-023	100	00:10:48
C-SACM-024	45	00:12:16	C-GSN-024	100	00:08:30
C-SACM-025	30	00:12:16	C-GSN-025	80.95	00:08:30
C-SACM-026	85	00:10:49	C-GSN-026	100	00:08:30
C-SACM-027	85	00:10:49	C-GSN-027	35.71	00:08:30
C-SACM-028	75	00:10:30	C-GSN-028	64.29	00:08:30
C-SACM-029	85	00:10:05	C-GSN-029	35.71	00:08:30
C-SACM-030	22.5	00:09:40	C-GSN-030	40.48	00:08:30
C-SACM-031	52.5	00:09:30	C-GSN-031	21.43	00:08:30
C-SACM-032	15	00:09:30	C-GSN-032	83.33	00:07:59
C-SACM-033	85	00:09:05	C-GSN-033	100	00:06:10
C-SACM-034	55	00:08:40	C-GSN-034	80.95	00:06:10
C-SACM-035	85	00:07:27	C-GSN-035	100	00:06:00
C-SACM-036	87.5	00:07:50	C-GSN-036	100	00:05:40

TABLE B.2: Scoring and timing result Group B: SACM notation and GSN

Group B					
SACM notation			GSN		
Participant ID	Score	Timing	Participant ID	Score	Timing
D-SACM-001	100	00:05:16	D-GSN-001	90.48	00:06:08
D-SACM-002	77.5	00:06:24	D-GSN-002	85.71	00:06:29
D-SACM-003	75	00:06:47	D-GSN-003	100	00:07:03
D-SACM-004	55	00:07:17	D-GSN-004	54.76	00:07:14
D-SACM-005	95	00:07:29	D-GSN-005	97.62	00:07:32
D-SACM-006	82.5	00:07:40	D-GSN-006	100	00:08:50
D-SACM-007	75	00:07:46	D-GSN-007	90.48	00:08:53
D-SACM-008	65	00:15:44	D-GSN-008	69.05	00:11:50
D-SACM-009	90	00:14:57	D-GSN-009	57.14	00:09:37
D-SACM-010	100	00:13:54	D-GSN-010	54.76	00:09:10
D-SACM-011	22.5	00:13:34	D-GSN-011	66.67	00:17:50
D-SACM-012	20	00:13:11	D-GSN-012	19.05	00:19:05
D-SACM-013	90	00:12:44	D-GSN-013	19.05	00:19:05
D-SACM-014	85	00:12:26	D-GSN-014	30.95	00:16:38
D-SACM-015	30	00:12:00	D-GSN-015	80.95	00:16:20
D-SACM-016	100	00:11:33	D-GSN-016	100	00:16:15
D-SACM-017	57.5	00:11:09	D-GSN-017	52.38	00:16:02
D-SACM-018	85	00:10:45	D-GSN-018	100	00:16:00
D-SACM-019	35	00:10:43	D-GSN-019	28.57	00:16:00
D-SACM-020	17.5	00:10:28	D-GSN-020	28.57	00:14:45
D-SACM-021	35	00:09:54	D-GSN-021	47.62	00:16:00
D-SACM-022	22.5	00:09:54	D-GSN-022	23.81	00:14:45
D-SACM-023	20	00:09:59	D-GSN-023	9.52	00:14:45
D-SACM-024	15	00:09:59	D-GSN-024	47.62	00:13:35
D-SACM-025	85	00:09:29	D-GSN-025	47.62	00:13:05
D-SACM-026	85	00:09:17	D-GSN-026	100	00:12:17
D-SACM-027	72.5	00:09:12	D-GSN-027	100	00:10:45
D-SACM-028	22.5	00:07:59	D-GSN-028	100	00:12:15
D-SACM-029	42.5	00:07:59	D-GSN-029	23.81	00:17:12
D-SACM-030	27.5	00:07:35	D-GSN-030	14.29	00:17:05
D-SACM-031	27.5	00:07:35	D-GSN-031	11.90	00:15:18
D-SACM-032	47.5	00:07:35	D-GSN-032	47.62	00:14:23
			D-GSN-033	76.19	00:11:43

B.3 Result of each element type (%)

TABLE B.3: Percentage of participants who accurately draw a particular type of element of SACM notation and GSN in their model

Group A SACM notation			
Element	Visual	Position	Both
Claim	67%	67%	58%
ArgumentReasoning	53%	64%	42%
ArtifactReference (context)	14%	81%	11%
ArtifactReference (evidence)	19%	78%	19%
AssertedContext	36%	75%	36%
AssertedInference	33%	56%	31%
AssertedEvidence	39%	72%	39%

Group A GSN			
Element	Visual	Position	Both
Goal	72%	78%	67%
Strategy	67%	75%	58%
Context	67%	83%	67%
Solution	67%	81%	64%
InContextOf	61%	83%	61%
SupportedBy	44%	56%	44%

Group B SACM notation			
Element	Visual	Position	Both
Claim	81%	47%	47%
ArgumentReasoning	56%	47%	44%
ArtifactReference (context)	19%	63%	19%
ArtifactReference (evidence)	25%	59%	22%
AssertedContext	41%	63%	41%
AssertedInference	34%	47%	34%
AssertedEvidence	41%	59%	41%

Group B GSN			
Element	Visual	Position	Both
Goal	79%	55%	48%
Strategy	67%	61%	55%
Context	48%	39%	36%
Solution	52%	55%	36%
InContextOf	42%	39%	36%
SupportedBy	27%	42%	24%

The result presented in Table B.3 shows the percentage of participants who correctly draw a particular

type of elements. For example, 67% of participants correctly draw the visual of all Claim's elements (G1, G3, G7, G8), 67% correctly drawn the position of all Claim's elements, and 58% participants correctly draw both visual and position of the elements.

B.4 Error identified of each element

B.4.1 Group A: GSN

This subsection presents mistakes committed by Group A participants in creating an assurance case model using GSN: Goal (Table B.4), Context (Table B.5), Strategy (Table B.6), Solution (Table B.7), SupportedBy (Table B.8), and InContextOf (Table B.9).

TABLE B.4: Mistakes committed by Group A participants: GSN Goals

Goal	% of participants who correctly drawn the element			Error found	
	Visual	Position	Both visual and position	Visual	Position
G1	86%	100%	86%	> Forgot VR (2) > Drawn as a Context (3)	N/A
G3	81%	94%	75%	> Forgot VR (1) > Drawn as a Strategy (2) > Drawn as a Solution (1) > Drawn as a Context (3)	Wrong place (2)
G7	75%	81%	69%	> No drawing (3) > Forgot VR (1) > Drawn as a reversed Strategy (1) > Drawn as a Context (4)	> Wrong place (4) > No drawing (3)
G8	75%	81%	69%	> No drawing (3) > Forgot VR (1) > Drawn as a reversed Strategy (1) > Drawn as a Context (4)	> Wrong place (4) > No drawing (3)

TABLE B.5: Mistakes committed by Group A participants: GSN Context

Context	% of participants who correctly drawn the element			Error found	
	Visual	Position	Both visual and position	Visual	Position
C1	81%	94%	81%	> Drawn as a rectangle (2) > Forgot VR (4) > Drawn as a Solution (1)	Wrong place (2)
C2	81%	94%	81%	> Drawn as a rectangle (2) > Forgot VR (4) > Drawn as a Solution (1)	Wrong place (2)
C4	67%	83%	67%	> Drawn as a rectangle (2) > Drawn as a circle (2) > No drawing (2) > Forgot VR (5) > Drawn as a Solution (1)	> Wrong place (3) > No drawing (2) > Connected to relationship between G3 and G7, G8 (1)
C5	67%	83%	67%	> Drawn as a rectangle (2) > Drawn as a circle (2) > No drawing (2) > Forgot VR (5) > Drawn as a Solution (1)	> Wrong place (3) > No drawing (2) > Connected to relationship between G3 and G7, G8 (1)

TABLE B.6: Mistakes committed by Group A participants: GSN Strategy

Strategy	% of participants who correctly drawn the element			Error found	
	Visual	Position	Both visual and position	Visual	Position
	67%	75%	58%	> Drawn as a rectangle (5) > No drawing (4) > Drawn as a Solution (1) > Drawn as a Context (1) > Drawn as SACM ArgumentReasoning (1)	> Wrong place (3) > No drawing (4) > Connected to relationship between G3 and G7, G8 (2)

TABLE B.7: Mistakes committed by Group A participants: GSN Solution

Solution	% of participants who correctly drawn the element			Error found	
	Visual	Position	Both visual and position	Visual	Position
Sn3	67%	81%	64%	> No drawing (5) > Forgot VR (4) > Drawn as a Strategy (1) > Drawn as a Context (2)	> Wrong place (2) > No drawing (5)
Sn4	67%	83%	67%	> No drawing (5) > Forgot VR (4) > Drawn as a Strategy (1) > Drawn as a Context (2)	> Wrong place (1) > No drawing (5)

TABLE B.8: Mistakes committed by Group A participants: GSN SupportedBy

SupportedBy	% of participants who correctly drawn the element			Error found	
	Visual	Position	Both visual and position	Visual	Position
G1 to G3	61%	94%	61%	> No drawing (1) > No arrowhead (6) > Wrong arrowhead (6) > Wrong direction (3)	> Wrong place (1) > No drawing (1)
G3 to S2	58%	69%	56%	> No drawing (7) > No arrowhead (5) > Wrong arrowhead (2) > Wrong direction (3)	> Wrong place (5) > No drawing (7)
S2 to G7	56%	67%	56%	> Drawn as InContextOf (1) > No drawing (7) > No arrowhead (4) > Wrong arrowhead (4) > Wrong direction (1)	> Wrong place (5) > No drawing (7)
S2 to G8	56%	67%	56%	> Drawn as InContextOf (1) > No drawing (6) > No arrowhead (4) > Wrong arrowhead (5) > Wrong direction (2)	> Wrong place (6) > No drawing (6)
G7 to Sn3	58%	78%	58%	> No drawing (5) > No arrowhead (4) > Wrong arrowhead (5) > Wrong direction (2)	> Wrong place (3) > No drawing (5)
G8 to Sn4	61%	81%	58%	> No drawing (4) > No arrowhead (4) > Wrong arrowhead (5) > Wrong direction (2)	> Wrong place (3) > No drawing (4)

B.4.2 Group A: SACM notation

This subsection presents mistakes committed by Group A participants in creating an assurance case model using SACM notation: Claim (Table B.10), ArgumentReasoning (Table B.11), ArtifactReference (Table B.12), AssertedContext (Table B.13), AssertedInference (Table B.14), and AssertedEvidence (Table B.15).

TABLE B.9: Mistakes committed by Group A participants: GSN InContextOf

InContextOf	% of participants who correctly drawn the element			Error found	
	Visual	Position	Both visual and position	Visual	Position
G1 to C1	64%	94%	64%	> No drawing (1) > No arrowhead (6) > Wrong arrowhead (4) > Wrong direction (2)	> Wrong place (1) > No drawing (1)
G1 to C2	64%	94%	64%	> No drawing (1) > No arrowhead (6) > Wrong arrowhead (4) > Wrong direction (2)	> Wrong place (1) > No drawing (1)
G3 to C4	64%	83%	64%	> Drawn as SupportedBy (1) > No drawing (3) > No arrowhead (6) > Wrong arrowhead (1) > Wrong direction (2)	> Wrong place (2) > No drawing (3) > Connected to relationship between G3 and G7, G8 (1)
G3 to C5	64%	83%	64%	> Drawn as SupportedBy (1) > No drawing (3) > No arrowhead (6) > Wrong arrowhead (1) > Wrong direction (2)	> Wrong place (2) > No drawing (3) > Connected to relationship between G3 and G7, G8 (1)

TABLE B.10: Mistakes committed by Group A participants: SACM notation Claim

Claim	% of participants who correctly drawn the element			Error found	
	Visual	Position	Both visual and position	Visual	Position
G1	92%	97%	92%	Forgot VR (2)	N/A
G3	83%	86%	78%	> Forgot VR (3) > Drawn as ArtifactReference (2)	Wrong place (4)
G7	67%	69%	61%	> Forgot VR (4) > No drawing (6) > Drawn as ArtifactReference (1)	> No drawing (6) > Wrong place (4)
G8	64%	67%	58%	> Forgot VR (4) > No drawing (7) > Drawn as ArtifactReference (1)	> No drawing (7) > Wrong place (4)

TABLE B.11: Mistakes committed by Group A participants: SACM notation ArgumentReasoning

ArgumentReasoning	% of participants who correctly drawn the element			Error found	
	Visual	Position	Both visual and position	Visual	Position
	50%	61%	39%	> Forgot VR (5) > No drawing (4) > Drawn as ArtifactReference (1) > Drawn as a rectangle (7)	> No drawing (4) > Wrong place (9)

B.4.3 Group B: GSN

This subsection presents mistakes committed by Group A participants in creating an assurance case model using GSN: Goal (Table B.16), Context (Table B.17), Strategy (B.18), Solution (Table B.19), SupportedBy (Table B.20), and InContextOf (Table B.21).

TABLE B.12: Mistakes committed by Group A participants: SACM notation ArtifactReference

ArtifactReference	% of participants who correctly drawn the element			Error found	
	Visual	Position	Both visual and position	Visual	Position
C1	11%	86%	11%	> Forgot VR (9) > No arrow (11) > Drawn as a rectangle (11)	Wrong place (4)
C2	11%	83%	11%	> Forgot VR (9) > No drawing (1) > No arrow (11) > Drawn as a rectangle (10)	> No drawing (1) > Wrong place (4)
C4	11%	78%	8%	> Forgot VR (9) > No drawing (1) > No arrow (11) > Drawn as a rectangle (10)	> No drawing (1) > Wrong place (6)
C5	11%	78%	8%	> Forgot VR (9) > No drawing (2) > No arrow (11) > Drawn as a rectangle (9)	> No drawing (2) > Wrong place (5)
Sn3	17%	75%	17%	> Forgot VR (6) > No drawing (6) > No arrow (10) > Drawn as a rectangle (7)	> No drawing (6) > Wrong place (2)
Sn4	17%	75%	17%	> Forgot VR (6) > No drawing (6) > No arrow (10) > Drawn as a rectangle (7)	> No drawing (6) > Wrong place (2)

TABLE B.13: Mistakes committed by Group A participants: SACM notation AssertedContext

AssertedContext	% of participants who correctly drawn the element			Error found	
	Visual	Position	Both visual and position	Visual	Position
C1 to G1	39%	81%	36%	> No drawing (1) > Wrong direction (6) > No dot (7) > No squarehead (2) > Drawn as an AssertedInference/Evidence (2) > Use arrowhead (7) > Drawn as a hollow squarehead (1) > Dot is drawn as a square (5)	> No drawing (1) > Wrong place (5)
C2 to G1	39%	81%	36%	> No drawing (1) > Wrong direction (6) > No dot (6) > No squarehead (3) > Drawn as an AssertedInference/Evidence (2) > Use arrowhead (6) > Drawn as a hollow squarehead (1) > Dot is drawn as a square (5)	> No drawing (1) > Wrong place (5)
C4 to G3	33%	75%	33%	> No drawing (1) > Wrong direction (6) > No dot (8) > No squarehead (4) > Drawn as an AssertedInference/Evidence (4) > Use arrowhead (6) > Drawn as a hollow squarehead (1) > Dot is drawn as a square (4)	> No drawing (1) > Wrong place (7)
C5 to G3	33%	75%	33%	> No drawing (2) > Wrong direction (6) > No dot (7) > No squarehead (4) > Drawn as an AssertedInference/Evidence (4) > Use arrowhead (5) > Drawn as a hollow squarehead (1) > Dot is drawn as a square (4)	> No drawing (2) > Wrong place (6)

TABLE B.14: Mistakes committed by Group A participants: SACM notation AssertedInference

AssertedInference	% of participants who correctly drawn the element			Error found	
	Visual	Position	Both visual and position	Visual	Position
G3 to G1	42%	86%	42%	<ul style="list-style-type: none"> > Wrong direction (7) > No dot (8) > Dot is drawn as a squarehead (2) > Wrong arrowhead (8) > No arrowhead (6) > Use hollow arrowhead (1) > Use dashed line (1) 	Wrong place (4)
G7 to G3	31%	58%	28%	<ul style="list-style-type: none"> > No drawing (6) > Wrong direction (3) > No dot (6) > Dot is drawn as a squarehead (1) > Wrong arrowhead (6) > No arrowhead (7) > Drawn as an AssertedContext (2) > Use hollow arrowhead (1) 	<ul style="list-style-type: none"> > No drawing (6) > Wrong place (8)
G8 to G3	31%	53%	28%	<ul style="list-style-type: none"> > No drawing (7) > Wrong direction (2) > No dot (6) > Dot is drawn as a squarehead (1) > Wrong arrowhead (4) > No arrowhead (8) > Drawn as an AssertedContext (2) > Use hollow arrowhead (1) 	<ul style="list-style-type: none"> > No drawing (7) > Wrong place (9)

TABLE B.15: Mistakes committed by Group A: SACM notation AssertedEvidence

AssertedEvidence	% of participants who correctly drawn the element			Error found	
	Visual	Position	Both visual and position	Visual	Position
Sn3 to G7	36%	72%	36%	<ul style="list-style-type: none"> > No drawing (7) > Wrong direction (2) > No dot (9) > Dot is drawn as a squarehead (1) > Wrong arrowhead (5) > No arrowhead (5) > Drawn as an AssertedContext (1) > Drawn as an AssertedInference/Evidence (1) 	<ul style="list-style-type: none"> > No drawing (7) > Wrong place (2)
Sn4 to G8	36%	69%	36%	<ul style="list-style-type: none"> > No drawing (7) > Wrong direction (2) > No dot (9) > Dot is drawn as a squarehead (1) > Wrong arrowhead (5) > No arrowhead (5) > Drawn as an AssertedContext (1) > Drawn as an AssertedInference/Evidence (1) 	<ul style="list-style-type: none"> > No drawing (7) > Wrong place (3)

B.4.4 Group B: SACM notation

This subsection presents mistakes committed by Group A participants in creating an assurance case model using SACM notation: Claim (Table B.22), ArgumentReasoning (Table B.24), ArtifactReference (Table B.23), AssertedContext (Table B.25), AssertedInference (Table B.26), and AssertedEvidence (Table B.27).

TABLE B.16: Mistakes committed by Group B participants: GSN Goals

Goal	% of participants who correctly drawn the element			Error found	
	Visual	Position	Both visual and position	Visual	Position
G1	94%	91%	85%	> Forgot VR (1) > Drawn as a Strategy (1)	Wrong place(3)
G3	88%	70%	61%	> Forgot VR (1) > Drawn as a Solution (2) > Drawn as a Context (1)	Wrong place (10)
G7	85%	82%	67%	> Drawn as a rectangle (1) > Drawn as a Solution (1) > Drawn as a Context (3)	Wrong place (6)
G8	85%	70%	64%	> Drawn as a rectangle (1) > Drawn as a Solution (3) > Drawn as a Context (1)	Wrong place (10)

TABLE B.17: Mistakes committed by Group B participants: GSN Context

Context	% of participants who correctly drawn the element			Error found	
	Visual	Position	Both visual and position	Visual	Position
C1	61%	79%	58%	> Drawn as a rectangle (7) > Forgot VR (2) > Drawn as a reversed Strategy (1) > Drawn as a Strategy (3)	Wrong place (7)
C2	64%	70%	58%	> Drawn as a rectangle (8) > Forgot VR (1) > Drawn as a reversed Strategy (1) > Drawn as a Strategy (2)	Wrong place (10)
C4	48%	39%	36%	> Drawn as a rectangle (7) > No drawing (6) > Forgot VR (2) > Drawn as a Strategy (2)	> Wrong place (14) > No drawing (6)
C5	48%	39%	36%	> Drawn as a rectangle (7) > No drawing (5) > Forgot VR (2) > Drawn as a Strategy (3)	> Wrong place (15) > No drawing (5)

TABLE B.18: Mistakes committed by Group B participants: GSN ArgumentReasoning

ArgumentReasoning	% of participants who correctly drawn the element			Error found	
	Visual	Position	Both visual and position	Visual	Position
	67%	61%	55%	> Forgot VR (2) > Drawn as a rectangle (8) > Drawn as a Solution (1)	Wrong place (13)

TABLE B.19: Mistakes committed by Group B participants: GSN Solution

Solution	% of participants who correctly drawn the element			Error found	
	Visual	Position	Both visual and position	Visual	Position
Sn3	52%	64%	36%	> Drawn as a rectangle (10) > No drawing (1) > Forgot VR (3) > Drawn as a Strategy (1) > Drawn as a Context (1)	> Wrong place (11) > No drawing (1)
Sn4	52%	61%	42%	> Drawn as a rectangle (11) > No drawing (1) > Forgot VR (2) > Drawn as a Strategy (1) > Drawn as a Context (1)	> Wrong place (12) > No drawing (1)

B.5 Identification of element variants result

This section presents the result of the identification of element variants of SACM notation and GSN.

TABLE B.20: Mistakes committed by Group B participants: GSN SupportedBy

SupportedBy	% of participants who correctly drawn the element			Error found	
	Visual	Position	Both visual and position	Visual	Position
G1 to G3	45%	61%	39%	> No drawing (10) > No arrowhead (1) > Wrong arrowhead (6) > Wrong direction (1)	> Wrong place (3) > No drawing (10)
G3 to S2	42%	61%	36%	> No drawing (11) > No arrowhead (1) > Wrong arrowhead (6) > Wrong direction (1)	> Wrong place (2) > No drawing (11)
S2 to G7	42%	79%	42%	> No drawing (7) > No arrowhead (6) > Wrong arrowhead (5) > Wrong direction (1)	No drawing (7)
S2 to G8	36%	64%	36%	> No drawing (10) > No arrowhead (5) > Wrong arrowhead (5) > Wrong direction (1)	> Wrong place (2) > No drawing (10)
G7 to Sn3	39%	64%	36%	> No drawing (8) > No arrowhead (7) > Wrong arrowhead (4) > Wrong direction (1)	> Wrong place (4) > No drawing (8)
G8 to Sn4	39%	58%	36%	> No drawing (9) > No arrowhead (4) > Wrong arrowhead (6) > Wrong direction (1)	> Wrong place (5) > No drawing (9)

TABLE B.21: Mistakes committed by Group B participants: GSN InContextOf

InContextOf	% of participants who correctly drawn the element			Error found	
	Visual	Position	Both visual and position	Visual	Position
G1 to C1	52%	79%	52%	> No drawing (1) > No arrowhead (8) > Wrong arrowhead (7) > Wrong direction (1)	> Wrong place (6) > No drawing (1)
G1 to C2	48%	70%	48%	> No drawing (7) > No arrowhead (4) > Wrong arrowhead (6)	> Wrong place (3) > No drawing (7)
G3 to C4	42%	39%	36%	> No drawing (10) > No arrowhead (4) > Wrong arrowhead (5) > Wrong direction (1)	> Wrong place (10) > No drawing (10)
G3 to C5	42%	45%	36%	> No drawing (13) > No arrowhead (3) > Wrong arrowhead (3)	> Wrong place (5) > No drawing (13)

TABLE B.22: Mistakes committed by Group B participants: SACM notation Claim

Claim	% of participants who correctly drawn the element			Error found	
	Visual	Position	Both visual and position	Visual	Position
G1	88%	84%	81%	> Forgot VR (1) > No drawing (2) > Drawn as an ArtifactReference (1)	> No drawing (2) > Wrong place (3)
G3	97%	88%	84%	Drawn as an ArtifactReference (1)	Wrong place (4)
G7	97%	66%	66%	Drawn as an ArtifactReference (1)	Wrong place (11)
G8	94%	50%	50%	> No drawing (1) > Drawn as an ArtifactReference (1)	> No drawing (1) > Wrong place (15)

B.5.1 GSN elements

This subsection presents the result of the identification of element variants of GSN.

TABLE B.23: Mistakes committed by Group B participants: SACM notation ArtifactReference

ArtifactReference	% of participants who correctly drawn the element			Error found	
	Visual	Position	Both visual and position	Visual	Position
C1	25%	84%	25%	> Forgot VR (2) > No drawing (2) > No arrow (5) > Drawn as a rectangle (13) > Drawn as a Context (2)	> No drawing (2) > Wrong place (3)
C2	25%	81%	25%	> Forgot VR (2) > No arrow (5) > Drawn as a rectangle (15) > Drawn as a Context (2)	Wrong place (6)
C4	19%	66%	19%	> Forgot VR (2) > No drawing (2) > No arrow (6) > Drawn as a rectangle (14) > Drawn as a Context (2)	> No drawing (2) > Wrong place (9)
C5	19%	63%	19%	> Forgot VR (2) > No drawing (2) > No arrow (6) > Drawn as a rectangle (13) > Drawn as a Context (2)	> No drawing (3) > Wrong place (9)
Sn3	25%	63%	22%	> Forgot VR (2) > No drawing (1) > No arrow (7) > Drawn as a rectangle (12) > Drawn as a Context (2)	> No drawing (1) > Wrong place (11)
Sn4	25%	63%	25%	> Forgot VR (2) > No drawing (1) > No arrow (7) > Drawn as a rectangle (12) > Drawn as a Context (2)	> No drawing (1) > Wrong place (11)

TABLE B.24: Mistakes committed by Group B participants: SACM notation ArgumentReasoning

ArgumentReasoning	% of participants who correctly drawn the element			Error found	
	Visual	Position	Both visual and position	Visual	Position
	56%	47%	44%	> No drawing (1) > Drawn as a rectangle (10) > Drawn as a Strategy (3)	> No drawing (1) > Wrong place (16)

TABLE B.25: Mistakes committed by Group B participants: SACM notation AssertedContext

AssertedContext	% of participants who correctly drawn the element			Error found	
	Visual	Position	Both visual and position	Visual	Position
C1 to G1	47%	88%	47%	> No drawing (2) > Wrong direction (10) > No dot (11) > No squarehead (4) > Wrong squarehead (9)	> No drawing (2) > Wrong place (2)
C2 to G1	47%	75%	47%	> No drawing (7) > Wrong direction (6) > No dot (6) > No squarehead (3) > Wrong squarehead (5)	> No drawing (7) > Wrong place (1)
C4 to G3	41%	66%	41%	> No drawing (8) > Wrong direction (8) > No dot (7) > No squarehead (2) > Wrong squarehead (6)	> No drawing (8) > Wrong place (3)
C5 to G3	41%	63%	41%	> No drawing (9) > Wrong direction (7) > No dot (6) > No squarehead (2) > Wrong squarehead (5)	> No drawing (9) > Wrong place (3)

TABLE B.26: Mistakes committed by Group B participants: SACM notation AssertedInference

AssertedInference	% of participants who correctly drawn the element			Error found	
	Visual	Position	Both visual and position	Visual	Position
G3 to G1	44%	81%	44%	> No drawing (6) > Wrong direction (2) > No dot (8) > Wrong arrowhead (6) > No arrowhead (4)	No drawing (6)
G7 to G3	41%	66%	41%	> No drawing (11) > No dot (4) > Wrong arrowhead (4) > No arrowhead (4)	No drawing (11)
G8 to G3	38%	50%	38%	> No drawing (16) > No dot (1) > Wrong arrowhead (3) > No arrowhead (1)	No drawing (16)

TABLE B.27: Mistakes committed by Group B participants: SACM notation AssertedEvidence

AssertedEvidence	% of participants who correctly drawn the element			Error found	
	Visual	Position	Both visual and position	Visual	Position
Sn3 to G7	44%	63%	44%	> No drawing (9) > Wrong direction (6) > No dot (6) > Wrong arrowhead (6) > No arrowhead (1)	> No drawing (9) > Wrong place (3)
Sn4 to G8	47%	66%	47%	> No drawing (8) > Wrong direction (4) > No dot (6) > Wrong arrowhead (7) > No arrowhead (1)	> No drawing (8) > Wrong place (3)

TABLE B.28: Identification of other types of GSN Goal element

Info given	Goal					
	Identify other types of Goal					
Question	Group A		Group B		Group A and B	
Answer option	49 participants		33 participants		Total 82 participants	
	Count	%	Count	%	Count	%
Assumption	3	6%	0	0%	3	4%
Justification	2	4%	2	6%	4	5%
Context	3	6%	1	3%	4	5%
Solution	4	8%	2	6%	6	7%
Strategy	4	8%	7	21%	11	13%
Module	10	20%	11	33%	21	26%
UndevelopedGoal	16	33%	9	27%	25	30%
UninstantiatedGoal	21	43%	11	33%	32	39%
Contract	8	16%	9	27%	17	21%
SupportedBy	2	4%	2	6%	4	5%
InContextOf	2	4%	2	6%	4	5%
AwayGoal	15	31%	17	52%	32	39%

B.5.2 SACM notation elements

This subsection presents the result of the identification of element variants of SACM notation.

TABLE B.29: Identification of GSN Module variant

Info given	Module					
Question	Identify other types of Module					
Answer option	Group A		Group B		Group A and B	
	49 participants		33 participants		Total 82 participants	
	Count	%	Count	%	Count	%
Assumption	1	2%	0	0%	1	1%
Justification	3	6%	0	0%	3	4%
Context	3	6%	0	0%	3	4%
Solution	1	2%	0	0%	1	1%
Strategy	3	6%	0	0%	3	4%
Goal	4	8%	5	15%	9	11%
UndevelopedGoal	7	14%	4	12%	11	13%
UninstantiatedGoal	3	6%	3	9%	6	7%
Contract	28	57%	25	76%	53	65%
SupportedBy	2	4%	0	0%	2	2%
InContextOf	1	2%	0	0%	1	1%
AwayGoal	12	24%	16	48%	28	34%

TABLE B.30: 'Claim' variants identification result

Info given	Claim					
Question	Identify Claim variants					
Answer option	Group A		Group B		Group A and B	
	49 participants		33 participants		Total 82 participants	
	Count	%	Count	%	Count	%
<i>ArtifactReference</i>	15	31%	10	30%	25	30%
<i>ArgumentPackage</i>	9	18%	4	12%	13	16%
<i>AssumedClaim</i>	11	22%	6	18%	17	21%
<i>AsCitedClaim</i>	11	22%	3	9%	14	17%
<i>AxiomaticClaim</i>	20	41%	13	39%	33	40%
<i>DefeatedClaim</i>	6	12%	1	3%	7	9%
<i>NeedsSupportClaim</i>	12	24%	5	15%	17	21%
<i>AbstractClaim</i>	13	27%	7	21%	20	24%
<i>ArgumentReasoning</i>	5	10%	3	9%	8	10%
<i>AssertedInference</i>	3	6%	4	12%	7	9%
<i>AssumedContext</i>	1	2%	0	0%	1	1%
<i>AxiomaticInference</i>	2	4%	0	0%	2	2%
<i>AssumedInference</i>	0	0%	0	0%	0	0%
<i>CounterInference</i>	0	0%	1	3%	1	1%
<i>MetaClaim</i>	2	4%	0	0%	2	2%
<i>ArgPkgBinding</i>	3	6%	4	12%	7	9%
<i>ArgPkgInterface</i>	1	2%	2	6%	3	4%

TABLE B.31: 'ArgumentPackage' variants identification result

Info given	ArgumentPackage					
Question	Identify ArgumentPackage variants					
Answer option	Group A		Group B		Group A and B	
	49 participants		33 participants		Total 82 participants	
	Count	%	Count	%	Count	%
<i>ArtifactReference</i>	6	12%	3	9%	9	11%
<i>Claim</i>	2	4%	1	3%	3	4%
<i>AssumedClaim</i>	1	2%	1	3%	2	2%
<i>AsCitedClaim</i>	5	10%	4	12%	9	11%
<i>AxiomaticClaim</i>	1	2%	1	3%	2	2%
<i>DefeatedClaim</i>	1	2%	3	9%	4	5%
<i>NeedsSupportClaim</i>	2	4%	2	6%	4	5%
<i>AbstractClaim</i>	0	0%	1	3%	1	1%
<i>ArgumentReasoning</i>	3	6%	5	15%	8	10%
<i>AssertedInference</i>	0	0%	0	0%	0	0%
<i>AssumedContext</i>	0	0%	0	0%	0	0%
<i>AxiomaticInference</i>	1	2%	0	0%	1	1%
<i>AssumedInference</i>	2	4%	0	0%	2	2%
<i>CounterInference</i>	0	0%	0	0%	0	0%
<i>MetaClaim</i>	1	2%	0	0%	1	1%
<i>ArgPkgBinding</i>	31	63%	18	55%	49	60%
<i>ArgPkgInterface</i>	28	57%	18	55%	46	56%

TABLE B.32: 'Assumed' element variants identification result

Info given	Assumed Claim					
Question	Identify other assumed elements					
Answer option	Group A		Group B		Group A and B	
	49 participants		33 participants		Total 82 participants	
	Count	%	Count	%	Count	%
<i>ArtifactReference</i>	3	6%	0	0%	3	4%
<i>Claim</i>	2	4%	1	3%	3	4%
<i>AsCitedClaim</i>	4	8%	10	30%	14	17%
<i>AxiomaticClaim</i>	8	16%	2	6%	10	12%
<i>DefeatedClaim</i>	5	10%	3	9%	8	10%
<i>NeedsSupportClaim</i>	24	49%	19	58%	43	52%
<i>AbstractClaim</i>	11	22%	7	21%	18	22%
<i>ArgumentReasoning</i>	2	4%	4	12%	6	7%
<i>AssertedInference</i>	1	2%	0	0%	1	1%
<i>AssumedContext</i>	8	16%	7	21%	15	18%
<i>AxiomaticInference</i>	3	6%	6	18%	9	11%
<i>AssumedInference</i>	12	24%	7	21%	19	23%
<i>CounterInference</i>	1	2%	0	0%	1	1%
<i>MetaClaim</i>	0	0%	1	3%	1	1%
<i>ArgPkgBinding</i>	0	0%	1	3%	1	1%
<i>ArgPkgInterface</i>	0	0%	0	0%	0	0%

TABLE B.33: ‘Axiomatic’ element variants identification result

Info given	Axiomatic Inference					
Question	Identify other Axiomatic elements					
Answer option	Group A		Group B		Group A and B	
	49 participants		33 participants		Total 82 participants	
	Count	%	Count	%	Count	%
<i>ArtifactReference</i>	4	8%	3	9%	7	9%
<i>Claim</i>	2	4%	3	9%	5	6%
<i>AsCitedClaim</i>	2	4%	0	0%	2	2%
<i>AxiomaticClaim</i>	5	10%	1	3%	6	7%
<i>DefeatedClaim</i>	2	4%	0	0%	2	2%
<i>NeedsSupportClaim</i>	3	6%	0	0%	3	4%
<i>AbstractClaim</i>	1	2%	2	6%	3	4%
<i>ArgumentReasoning</i>	3	6%	2	6%	5	6%
<i>AssertedInference</i>	13	27%	11	33%	24	29%
<i>AssumedContext</i>	3	6%	6	18%	9	11%
<i>AxiomaticContext</i>	23	47%	19	58%	42	51%
<i>AssumedInference</i>	8	16%	4	12%	12	15%
<i>CounterInference</i>	10	20%	13	39%	23	28%
<i>MetaClaim</i>	1	2%	4	12%	5	6%
<i>ArgPkgBinding</i>	2	4%	1	3%	3	4%
<i>ArgPkgInterface</i>	0	0%	0	0%	0	0%

TABLE B.34: ‘Defeated’ element variants identification result

Info given	Defeated Claim					
Question	Identify other Defeated elements					
Answer option	Group A		Group B		Group A and B	
	49 participants		33 participants		Total 82 participants	
	Count	%	Count	%	Count	%
<i>ArtifactReference</i>	0	0%	1	3%	1	1%
<i>Claim</i>	3	6%	1	3%	4	5%
<i>AsCitedClaim</i>	7	14%	8	24%	15	18%
<i>AxiomaticClaim</i>	4	8%	5	15%	9	11%
<i>DefeatedInference</i>	24	49%	10	30%	34	41%
<i>NeedsSupportClaim</i>	2	4%	2	6%	4	5%
<i>AbstractClaim</i>	5	10%	10	30%	15	18%
<i>ArgumentReasoning</i>	1	2%	1	3%	2	2%
<i>AssertedInference</i>	2	4%	0	0%	2	2%
<i>DefeatedContext</i>	20	41%	14	42%	34	41%
<i>AxiomaticContext</i>	2	4%	1	3%	3	4%
<i>AssumedInference</i>	1	2%	1	3%	2	2%
<i>CounterInference</i>	0	0%	1	3%	1	1%
<i>MetaClaim</i>	0	0%	0	0%	0	0%
<i>ArgPkgBinding</i>	0	0%	1	3%	1	1%
<i>ArgPkgInterface</i>	1	2%	1	3%	2	2%

B.6 Visual variables utilisation result

Layout and arrow direction

There were four questions for the participants to answer related to the utilisation of Shape in the context of the intuitiveness of the relationship element. The objective of these questions was to investigate the role of the layout and arrow direction in term its intuitiveness when being used in a diagram.

Figure B.3 presents the first question that was asked to the participants. The participants’ responses are presented in Table B.38.

TABLE B.35: ‘Abstract’ element variants identification result

Info given	Abstract Claim					
Question	Identify other Abstract elements					
Answer option	Group A		Group B		Group A and B	
	49 participants		33 participants		Total 82 participants	
	Count	%	Count	%	Count	%
AbstractArtifactReference	29	59%	20	61%	49	60%
<i>Claim</i>	2	4%	0	0%	2	2%
<i>AsCitedClaim</i>	1	2%	5	15%	6	7%
<i>AxiomaticClaim</i>	2	4%	0	0%	2	2%
<i>DefeatedInference</i>	0	0%	0	0%	0	0%
<i>NeedsSupportClaim</i>	2	4%	3	9%	5	6%
AbstractInference	21	43%	17	52%	38	46%
<i>ArgumentReasoning</i>	1	2%	1	3%	2	2%
<i>AssertedInference</i>	0	0%	0	0%	0	0%
<i>DefeatedContext</i>	1	2%	1	3%	2	2%
AbstractContext	19	39%	11	33%	30	37%
<i>AssumedInference</i>	2	4%	1	3%	3	4%
<i>CounterInference</i>	0	0%	1	3%	1	1%
<i>MetaClaim</i>	1	2%	0	0%	1	1%
<i>ArgPkgBinding</i>	1	2%	1	3%	2	2%
<i>ArgPkgInterface</i>	1	2%	4	12%	5	6%

TABLE B.36: ‘AsCited’ element variants identification result

Info given	AsCited Claim					
Question	Identify other AsCited elements					
Answer option	Group A		Group B		Group A and B	
	49 participants		33 participants		Total 82 participants	
	Count	%	Count	%	Count	%
<i>AbstractArtifactReference</i>	3	6%	5	15%	8	10%
<i>Claim</i>	4	8%	1	3%	5	6%
AsCitedInference	24	49%	9	27%	33	40%
<i>AxiomaticClaim</i>	3	6%	11	33%	14	17%
<i>DefeatedInference</i>	0	0%	0	0%	0	0%
<i>NeedsSupportClaim</i>	4	8%	2	6%	6	7%
<i>AbstractInference</i>	0	0%	0	0%	0	0%
<i>ArgumentReasoning</i>	5	10%	3	9%	8	10%
<i>AssertedInference</i>	1	2%	0	0%	1	1%
<i>DefeatedContext</i>	1	2%	1	3%	2	2%
AsCitedContext	21	43%	10	30%	31	38%
<i>AssumedInference</i>	3	6%	0	0%	3	4%
<i>CounterInference</i>	2	4%	0	0%	2	2%
<i>MetaClaim</i>	0	0%	1	3%	1	1%
<i>ArgPkgBinding</i>	4	8%	5	15%	9	11%
<i>ArgPkgInterface</i>	4	8%	4	12%	8	10%

In the second question, a dark brightness arrowhead was added as the decoration of the relationship that is pointing to a particular Claim, as shown in Figure B.4.

The participants’ responses regarding the second question (Figure B.4) is presented in Table B.39.

For the third and fourth questions, the layout of the diagram example was changed into horizontal. This is due to investigate the effect of whether the position in the context of layout might affect the participants’ decision regarding the intuitiveness of a relationship element.

Figure B.5 illustrate the third question that was asked to the participants.

The participants’ responses regarding the third question (Figure B.5) is presented in Table B.40.

TABLE B.37: ‘NeedsSupport’ element variants identification result

Info given Question	NeedsSupport Inference					
	Identify other NeedsSupport elements					
	Group A 49 participants		Group B 33 participants		Group A and B Total 82 participants	
Answer option	Count	%	Count	%	Count	%
<i>AbstractArtifactReference</i>	0	0%	0	0%	0	0%
<i>Claim</i>	1	2%	1	3%	2	2%
<i>AsCitedInference</i>	1	2%	5	15%	6	7%
<i>AxiomaticClaim</i>	2	4%	1	3%	3	4%
<i>DefeatedInference</i>	0	0%	3	9%	3	4%
<i>NeedsSupportClaim</i>	29	59%	16	48%	45	55%
<i>AbstractInference</i>	3	6%	0	0%	3	4%
<i>ArgumentReasoning</i>	1	2%	1	3%	2	2%
<i>AssertedInference</i>	4	8%	3	9%	7	9%
<i>DefeatedContext</i>	3	6%	0	0%	3	4%
<i>NeedSupportContext</i>	24	49%	22	67%	46	56%
<i>AssumedInference</i>	3	6%	2	6%	5	6%
<i>CounterInference</i>	4	8%	5	15%	9	11%
<i>MetaClaim</i>	1	2%	1	3%	2	2%
<i>ArgPkgBinding</i>	3	6%	0	0%	3	4%
<i>ArgPkgInterface</i>	2	4%	0	0%	2	2%

If there are two claims in a diagram (Claim A and B), one as a premise and another is the conclusion.

According to the following diagram, in your opinion, which Claim indicates the conclusion claim?



FIGURE B.3: Role of arrow direction in representing relationship element: Question #1

TABLE B.38: Participants’ response regarding the first question related to the role of the arrowhead to represent relationship element

Answer option	A (40)		B (31)		Total A+B	
	Count	%	Count	%	Count	%
Claim A	11	28%	6	19%	17	24%
Claim B	9	23%	11	35%	20	28%
None of them	4	10%	5	16%	9	13%
Both of them	14	35%	8	26%	22	31%
No answer	2	5%	1	3%	3	4%

TABLE B.39: Participants’ response regarding the second question related to the role of the arrowhead to represent relationship element

Answer option	A (40)		B (31)		Total A+B	
	Count	%	Count	%	Count	%
Claim A	25	63%	18	58%	43	61%
Claim B	9	23%	11	35%	20	28%
None of them	3	8%	1	3%	4	6%
Both of them	1	3%	1	3%	2	3%
No answer	2	5%	0	0%	2	3%

If there are two claims in a diagram (Claim A and B), one as a premise and another is the conclusion.
According to the following diagram, in your opinion, which Claim indicates the conclusion claim?

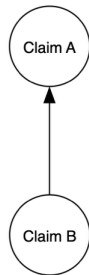


FIGURE B.4: Role of arrow direction in representing relationship element: Question #2

If there are two claims in a diagram (Claim A and B), one as a premise and another is the conclusion.
According to the following diagram, in your opinion, which Claim indicates the conclusion claim?



FIGURE B.5: Role of arrow direction in representing relationship element: Question #3

TABLE B.40: Participants' response regarding the third question related to the role of the arrowhead to represent relationship element

Answer option	A (40)		B (31)		Total A+B	
	Count	%	Count	%	Count	%
Claim A	2	5%	1	3%	3	4%
Claim B	5	13%	3	10%	8	11%
None of them	14	35%	15	48%	29	41%
Both of them	17	43%	12	39%	29	41%
No answer	2	5%	0	0%	2	3%

In the fourth question, a dark brightness arrowhead was added as the decoration of the relationship that is used to associate between "Claim A and B" as shown in Figure B.6.

If there are two claims in a diagram (Claim A and B), one as a premise and another is the conclusion.
According to the following diagram, in your opinion, which Claim indicates the conclusion claim?



FIGURE B.6: Role of arrow direction in representing relationship element: Question #4

The participants' responses regarding the question number four (Figure B.6) is presented in Table B.41.

Size in the context of line thickness

TABLE B.41: Participants' response regarding the fourth question related to the role of the arrowhead to represent relationship element

Answer option	A (40)		B (31)		Total A+B	
	Count	%	Count	%	Count	%
Claim A	8	20%	2	6%	10	14%
Claim B	26	65%	24	77%	50	70%
None of them	3	8%	4	13%	7	10%
Both of them	1	3%	1	3%	2	3%
No answer	2	5%	0	0%	2	3%

The next question that was asked to the participants is related to the utilisation of Size visual variable in the context of the line size thickness as used to represent the 'axiomatic' element in SACM notation. Figure B.7 illustrate the related question.

If there are two claims (Claim A and B) in a diagram that are rendered with different emphasise of the size of line thickness, in your opinion, which Claim indicates the conclusion claim?



FIGURE B.7: Role of line thickness in notation design

The participants responses regarding the question as shown in Figure B.7 is presented in Table B.42.

TABLE B.42: Participants' responses regarding the question related to the role of visual variable Size in the context of line thickness

Answer option	A (40)		B (31)		Total A+B	
	Count	%	Count	%	Count	%
Claim A	26	65%	19	61%	45	63%
Claim B	4	10%	4	13%	8	11%
None of them	6	15%	6	19%	12	17%
Both of them	2	5%	2	6%	4	6%
No answer	2	5%	0	0%	2	3%

Role of Brightness

In the last question, the participants were asked to provide their perception regarding the different use of visual variable brightness in notation design.

The participants were given a two different diagram; the first diagram presents Claim A vertically located relative to Claim B and these Claims were connected via a line with a dark brightness circle attach to Claim A. This diagram indicates that Claim A is supported by Claim B. In the second diagram, the position of Claim A and B is similar with the first diagram. The only difference is the brightness of the circle that is used as part of the relationship visual representation. The circle, in the second diagram, was drawn using dark brightness (as can be seen in Figure B.8). The participants were asked to provide their perception regarding this question.

Please take a look at the following diagram.



If the dark (brightness) circle-head line indicates a relationship that Claim A is supported by Claim B, in your opinion, what can be indicated by a light (brightness) circle-head line used as a relationship between Claim A and B as shown in the following diagram?

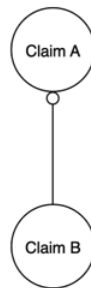


FIGURE B.8: Role of Brightness in notation design

The answer options provided to the user were considered the effect of the learning process, for example, related to the GSN notation. In this case, the dark and light brightness of particular elements such as used in the (InContextOf and SupportedBy) relationship. To be noted, the participants who involved in the experiment were not provided by the information related to the used of SACM ‘Counter’ relationship visual representation that is visually represented using a light brightness arrow-head line.

Table B.43 presents the participants’ responses related to this question.

TABLE B.43: Participants’ responses regarding the role of Brightness in the context of relationship visual representation design

Answer option	A (40)		B (31)		Total A+B	
	Count	%	Count	%	Count	%
Claim A is supported by Claim B	5	13%	5	16%	10	14%
Claim A is opposed by Claim B	4	10%	7	23%	11	15%
Claim A provide context to Claim B	10	25%	10	32%	20	28%
Claim B provide context to Claim A	13	33%	8	26%	21	30%
Nothing can be inferred	6	15%	0	0%	6	8%
Other	0	0%	1	3%	1	1%
No answer	2	5%	0	0%	2	3%

The participants’ responses regarding the question as shown in Figure B.8 shows that from total 71

participants, most of them (30%) agreed that the "Claim B" provide context to "Claim A". 28% participants also agreed that the light brightness circle head line provides context to a particular element especially in the case "Claim A" provide context to "Claim B".

The answer that we are expected from the participants was: "Claim A is opposed by Claim B". This is due to help us to confirm that if the dark brightness used to indicates "support" relationship, the light brightness should be (intuitively) indicates the opposite of "support". However, only 15% participants who chose this answer. We assumed that the learning effect, especially after learning the GSN (InContextOf) relationship, might affect the participants' responses regarding this question.

B.7 Participants' perception towards the notations

TABLE B.44: Participants' perception regarding the notation that is relatively easy to learn

Q:	Which notation do you considered as the notation that is relatively easy to learn?	Group A		Group B		Group A & B	
		40 participants		31 participants		71 participants	
		Count	%	Count	%	Total	%
A:	SACM notation	16	40%	6	19%	22	31%
	GSN	11	28%	16	52%	27	38%
	None of them	3	8%	0	0%	3	4%
	Both of them	8	20%	9	29%	17	24%
	No answer	2	5%	0	0%	2	3%

TABLE B.45: Participants' perception regarding the notation that is relatively easy to use

Q:	Which notation do you considered as the notation that is relatively easy to use?	Group A		Group B		Group A & B	
		40 participants		31 participants		71 participants	
		Count	%	Count	%	Total	%
A:	SACM notation	5	13%	8	26%	13	18%
	GSN	25	63%	18	58%	43	61%
	None of them	6	15%	1	3%	7	10%
	Both of them	2	5%	4	13%	6	8%
	No answer	2	5%	0	0%	2	3%

TABLE B.46: Participants' perception regarding the notation that they intend to use in the future

Q:	Which notation do you intend to use in practice when being asked to construct an assurance case?	Group A		Group B		Group A & B	
		40 participants		31 participants		71 participants	
		Count	%	Count	%	Total	%
A:	SACM notation	10	25%	14	45%	24	34%
	GSN	16	40%	14	45%	30	42%
	None of them	2	5%	0	0%	2	3%
	Both of them	10	25%	3	10%	13	18%
	No answer	2	5%	0	0%	2	3%

Appendix C

Alternate SACM notation

Figure C.1 illustrates the created visual representation of the Alternate SACM notation elements. These visual representations were created based on the proposed visual notation design approach with different design decisions that were used to create the SACM notation.

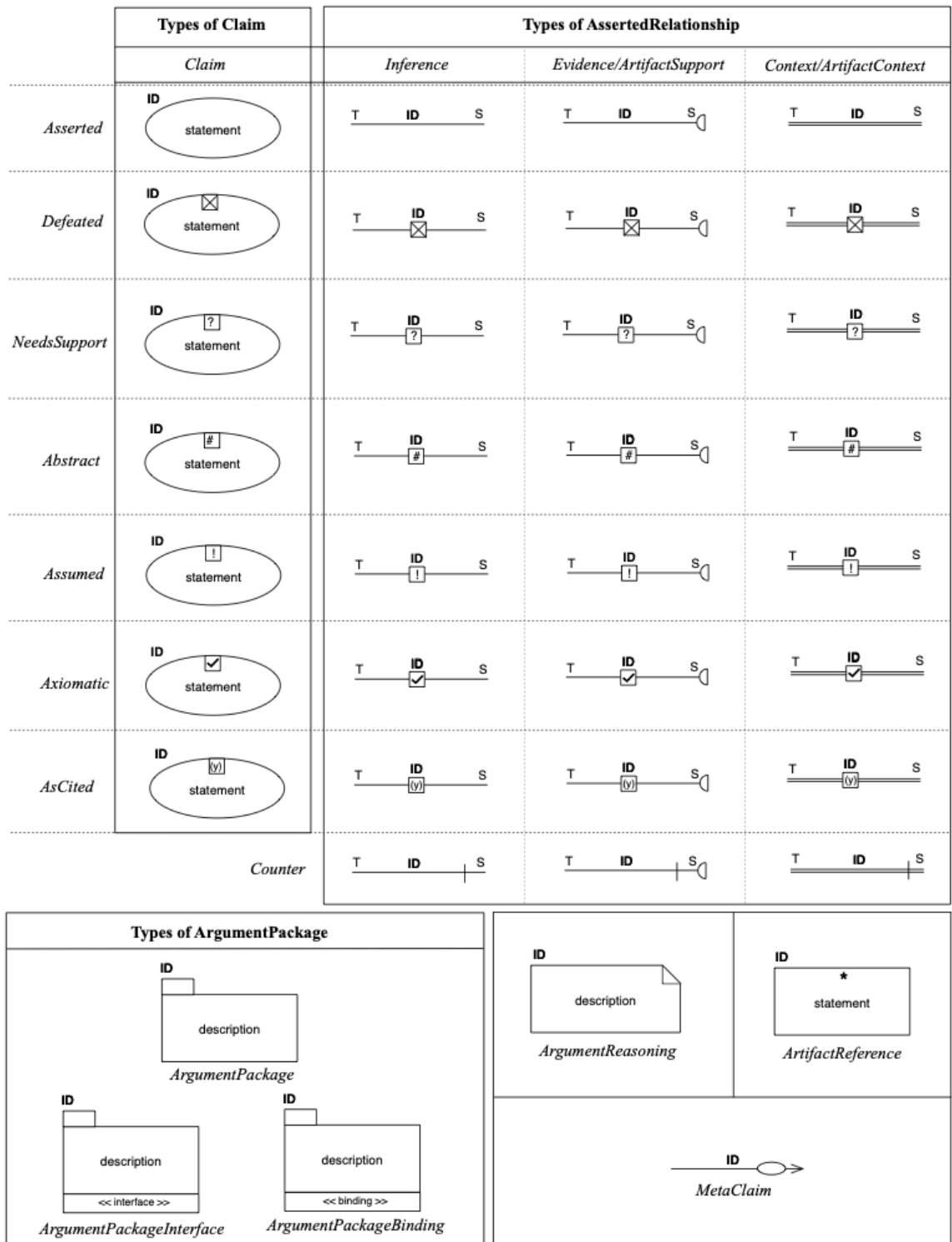


FIGURE C.1: Summary of the Alternate SACM notation

Appendix D

Empirical study data: SACM notation - SACM alternate notation

D.1 Assurance case diagram based on case study

Figure D.1 shows the assurance case presented using SACM alternate notation based on case study.

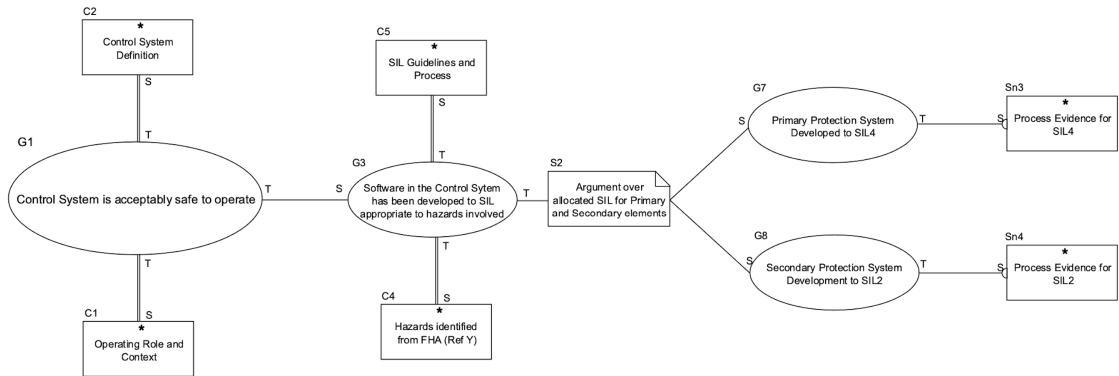


FIGURE D.1: Assurance case diagram constructed using SACM alternate notation based case study

D.2 Scoring and timing result: SACM notation and SACM alternate notation

Table D.1 presents Group A participants’ scoring and timing result in creating an assurance case using both SACM notation and SACM alternate notation, and Table D.2 for Group B participants.

TABLE D.1: Scoring and timing result Group A: SACM notation and SACM alternate notation

Group A					
SACM notation			SACM alternate notation		
Participant ID	Score	Timing	Participant ID	Timing	Score
A-SACM-001	67.5	20:51	A-SACMALT-001	16:12	72.5
A-SACM-002	62.5	20:49	A-SACMALT-002	16:11	97.5
A-SACM-003	65	20:27	A-SACMALT-003	16:07	80
A-SACM-004	27.5	20:41	A-SACMALT-004	15:59	97.5
A-SACM-005	70	20:28	A-SACMALT-005	15:57	67.5
A-SACM-006	67.5	20:15	A-SACMALT-006	15:50	90
A-SACM-007	75	20:08	A-SACMALT-007	15:31	62.5
A-SACM-008	95	17:29	A-SACMALT-008	15:27	77.5
A-SACM-009	85	16:37	A-SACMALT-009	15:20	97.5
A-SACM-010	95	16:29	A-SACMALT-010	13:22	30
A-SACM-011	17.5	16:19	A-SACMALT-011	13:00	97.5
A-SACM-012	95	16:06	A-SACMALT-012	12:32	92.5
A-SACM-013	40	15:31	A-SACMALT-013	10:48	87.5
A-SACM-014	55	15:21	A-SACMALT-014	10:44	70
A-SACM-015	100	14:45	A-SACMALT-015	9:44	17.5
A-SACM-016	77.5	14:20	A-SACMALT-016	8:40	97.5
A-SACM-017	90	13:54	A-SACMALT-017	8:24	77.5
A-SACM-018	85	13:05	A-SACMALT-018	8:18	97.5
A-SACM-019	47.5	12:27	A-SACMALT-019	8:04	92.5
A-SACM-020	65	11:32	A-SACMALT-020	7:46	97.5
A-SACM-021	100	09:08	A-SACMALT-021	7:33	92.5
A-SACM-022	100	07:28	A-SACMALT-022	6:52	97.5
A-SACM-023	100	08:08	A-SACMALT-023	6:10	92.5
A-SACM-024	100	05:36	A-SACMALT-024	6:10	80

TABLE D.2: Scoring and timing result Group B: SACM notation and SACM alternate notation

Group B					
SACM notation			SACM alternate notation		
Participant ID	Score	Timing	Participant ID	Score	Timing
B-SACM-001	95	05:25	B-SACMALT-001	90	07:05
B-SACM-002	85	06:04	B-SACMALT-002	57.5	08:50
B-SACM-003	42.5	06:07	B-SACMALT-003	52.5	08:54
B-SACM-004	95	07:04	B-SACMALT-004	55	09:09
B-SACM-005	85	07:19	B-SACMALT-005	80	09:19
B-SACM-006	57.5	07:55	B-SACMALT-006	82.5	09:19
B-SACM-007	82.5	08:15	B-SACMALT-007	20	10:40
B-SACM-008	20	08:18	B-SACMALT-008	37.5	10:42
B-SACM-009	27.5	08:36	B-SACMALT-009	57.5	11:05
B-SACM-010	27.5	08:37	B-SACMALT-010	55	11:14
B-SACM-011	100	08:49	B-SACMALT-011	57.5	11:16
B-SACM-012	67.5	08:53	B-SACMALT-012	95	11:22
B-SACM-013	52.5	09:00	B-SACMALT-013	70	11:43
B-SACM-014	40	09:03	B-SACMALT-014	90	11:46
B-SACM-015	12.5	09:10	B-SACMALT-015	90	11:47
B-SACM-016	42.5	09:13	B-SACMALT-016	95	11:49
B-SACM-017	97.5	09:16	B-SACMALT-017	77.5	11:52
B-SACM-018	85	09:26	B-SACMALT-018	95	11:57
B-SACM-019	47.5	09:29	B-SACMALT-019	72.5	11:58
B-SACM-020	70	09:35	B-SACMALT-020	60	12:04
B-SACM-021	62.5	09:47	B-SACMALT-021	65	12:12
B-SACM-022	100	09:49	B-SACMALT-022	67.5	12:13
B-SACM-023	100	10:26	B-SACMALT-023	65	12:20
B-SACM-024	37.5	10:29	B-SACMALT-024	62.5	12:25
B-SACM-025	72.5	10:37	B-SACMALT-025	62.5	13:03
B-SACM-026	87.5	10:41	B-SACMALT-026	52.5	13:28
B-SACM-027	20	11:05	B-SACMALT-027	95	14:02
B-SACM-028	80	11:07	B-SACMALT-028	87.5	14:31
B-SACM-029	20	11:09	B-SACMALT-029	62.5	14:44
B-SACM-030	15	11:09	B-SACMALT-030	40	15:30
B-SACM-031	70	11:58	B-SACMALT-031	37.5	16:20
B-SACM-032	100	12:01	B-SACMALT-032	25	16:24
B-SACM-033	72.5	12:18			
B-SACM-034	72.5	12:26			
B-SACM-035	75	12:28			

D.3 Result of each element type (%)

The result presented in Table D.3 shows the percentage of participants who correctly draw a particular type of elements. For example, 83% of participants correctly draw the visual of all Claim's elements (G1, G3, G7, G8), 79% correctly drawn the position of all Claim's elements, and 79% participants correctly draw both visual and position of the elements.

TABLE D.3: Percentage of participants who accurately draw a particular element type of SACM notation and SACM alternate notation in their model

Group A SACM notation				Group A SACM alternate notation			
Element	Visual	Position	Both	Element	Visual	Position	Both
Claim	83%	79%	79%	Claim	0%	75%	0%
ArgumentReasoning	71%	54%	54%	ArgumentReasoning	83%	67%	67%
ArtifactReference (context)	46%	83%	46%	ArtifactReference (context)	67%	92%	63%
ArtifactReference (evidence)	42%	83%	42%	ArtifactReference (evidence)	67%	88%	63%
AssertedContext	54%	83%	46%	AssertedContext	88%	92%	88%
AssertedInference	63%	79%	58%	AssertedInference	79%	58%	58%
AssertedEvidence	67%	83%	63%	AssertedEvidence	71%	83%	63%

Group B SACM notation				Group B SACM alternate notation			
Element	Visual	Position	Both	Element	Visual	Position	Both
Claim	80%	60%	51%	Claim	19%	72%	19%
ArgumentReasoning	66%	51%	49%	ArgumentReasoning	66%	72%	53%
ArtifactReference (context)	23%	69%	23%	ArtifactReference (context)	28%	78%	28%
ArtifactReference (evidence)	26%	77%	26%	ArtifactReference (evidence)	34%	100%	34%
AssertedContext	49%	66%	46%	AssertedContext	22%	75%	22%
AssertedInference	40%	51%	40%	AssertedInference	41%	66%	34%
AssertedEvidence	43%	77%	43%	AssertedEvidence	34%	100%	34%

D.4 Error identified of each element

D.4.1 Group A: SACM notation

This subsection presents mistakes committed by Group A participants in creating an assurance case model using SACM notation: Claim (Table D.4), ArgumentReasoning (Table D.5), ArtifactReference (Table D.6), AssertedContext (Table D.7), AssertedInference (Table D.8), and AssertedEvidence (Table D.9).

TABLE D.4: Mistakes committed by Group A participants: SACM notation Claim

Claim	% of participants who correctly drawn the element			Error found	
	Visual	Position	Both visual and position	Visual	Position
G1	100%	100%	100%	N/A	N/A
G3	92%	92%	88%	> Drawn as a rectangle (1) > Drawn as an ArtifactReference (1)	Wrong place (2)
G7	92%	79%	79%	> No drawing (1) > Drawn as a rectangle (1)	> No drawing (1) > Wrong place (2)
G8	88%	79%	79%	> No drawing (2) > Drawn as a rectangle (1)	> No drawing (2) > Wrong place (3)

TABLE D.5: Mistakes committed by Group A participants: SACM notation ArgumentReasoning

ArgumentReasoning	% of participants who correctly drawn the element			Error found	
	Visual	Position	Both visual and position	Visual	Position
	71%	54%	54%	> Drawn vertically (1) > Drawn as a rectangle (2) > Drawn as an ArtifactReference (3) > Wrong connector (1)	Wrong place (11)

TABLE D.6: Mistakes committed by Group A participants: SACM notation ArtifactReference

ArtifactReference	% of participants who correctly drawn the element			Error found	
	Visual	Position	Both visual and position	Visual	Position
C1	50%	92%	50%	> Forgot VR (2) > No arrow (3) > Drawn as a rectangle (7)	Wrong place (2)
C2	50%	92%	50%	> Forgot VR (2) > No arrow (3) > Drawn as a rectangle (7)	Wrong place (2)
C4	50%	83%	50%	> Forgot VR (2) > No arrow (3) > Drawn as a rectangle (7)	Wrong place (4)
C5	46%	83%	46%	> Forgot VR (2) > No arrow (4) > Drawn as a rectangle (7)	Wrong place (4)
Sn3	42%	88%	42%	> Forgot VR (3) > No drawing (2) > No arrow (6) > Drawn as a rectangle (3)	> Wrong place (1) > No drawing (2)
Sn4	42%	83%	42%	> Forgot VR (3) > No drawing (2) > No arrow (6) > Drawn as a rectangle (3)	> Wrong place (2) > No drawing (2)

TABLE D.7: Mistakes committed by Group A participants: SACM notation AssertedContext

AssertedContext	% of participants who correctly drawn the element			Error found	
	Visual	Position	Both visual and position	Visual	Position
C1 to G1	54%	92%	50%	> Wrong direction (9) > No dot (2) > Wrong squarehead (6)	Wrong place (2)
C2 to G1	54%	92%	50%	> Wrong direction (9) > No dot (2) > Wrong squarehead (6)	Wrong place (2)
C4 to G3	58%	83%	46%	> Wrong direction (9) > No dot (2) > Wrong squarehead (5)	Wrong place (4)
C5 to G3	58%	83%	46%	> Wrong direction (9) > No dot (2) > Wrong squarehead (5)	Wrong place (4)

TABLE D.8: Mistakes committed by Group A participants: SACM notation AssertedInference

AssertedInference	% of participants who correctly drawn the element			Error found	
	Visual	Position	Both visual and position	Visual	Position
G3 to G1	71%	88%	67%	> No drawing (1) > Wrong direction (1) > No dot (2) > Wrong arrowhead (5)	> No drawing (1) > Wrong place (2)
G7 to G3	67%	79%	58%	> No drawing (3) > No dot (2) > Wrong arrowhead (4) > no arrowhead (1)	> No drawing (3) > Wrong place (2)
G8 to G3	63%	79%	58%	> No drawing (4) > No dot (2) > Wrong arrowhead (4) > no arrowhead (1)	> No drawing (4) > Wrong place (1)

D.4.2 Group A: SACM alternate notation

This subsection presents mistakes committed by Group A participants in creating an assurance case model using SACM notation: Claim (Table D.10), ArgumentReasoning (Table D.11), ArtifactReference (Table D.12), AssertedContext (Table D.13), AssertedInference (Table D.14), and AssertedEvidence (Table D.15).

TABLE D.9: Mistakes committed by Group A participants: SACM notation AssertedEvidence

AssertedEvidence	% of participants who correctly drawn the element			Error found	
	Visual	Position	Both visual and position	Visual	Position
Sn3 to G7	67%	88%	67%	> No drawing (2) > No dot (2) > Wrong arrowhead (5) > no arrowhead (1)	> No drawing (2) > Wrong place (1)
Sn4 to G8	67%	83%	63%	> No drawing (2) > No dot (2) > Wrong arrowhead (5) > no arrowhead (1)	> No drawing (2) > Wrong place (2)

TABLE D.10: Mistakes committed by Group A participants: SACM alternate notation Claim

Claim	% of participants who correctly drawn the element			Error found	
	Visual	Position	Both visual and position	Visual	Position
G1	0%	100%	0%	> Not thick (22) > Not the biggest (15)	N/A
G3	96%	88%	88%	Drawn as a context (1)	Wrong place (3)
G7	96%	79%	79%	Drawn as a context (1)	Wrong place (5)
G8	96%	79%	79%	Drawn as a context (1)	Wrong place (5)

TABLE D.11: Mistakes committed by Group A participants: SACM alternate notation ArgumentReasoning

ArgumentReasoning	% of participants who correctly drawn the element			Error found	
	Visual	Position	Both visual and position	Visual	Position
	83%	67%	67%	> Forgot VR (1) > Drawn as a rectangle (3)	Wrong place (8)

TABLE D.12: Mistakes committed by Group A participants: SACM alternate notation ArtifactReference

ArtifactReference	% of participants who correctly drawn the element			Error found	
	Visual	Position	Both visual and position	Visual	Position
C1	88%	92%	83%	> No star (2) > Drawn as an ellipse (1)	Wrong place (2)
C2	88%	92%	83%	> No star (2) > Drawn as an ellipse (1)	Wrong place (2)
C4	79%	92%	71%	No star (5)	Wrong place (2)
C5	71%	92%	63%	No star (7)	Wrong place (2)
Sn3	67%	88%	63%	> No star (4) > Drawn as an ellipse (4)	Wrong place (3)
Sn4	67%	88%	63%	> No star (4) > Drawn as an ellipse (4)	Wrong place (3)

TABLE D.13: Mistakes committed by Group A participants: SACM alternate notation AssertedContext

AssertedContext	% of participants who correctly drawn the element			Error found	
	Visual	Position	Both visual and position	Visual	Position
C1 to G1	88%	92%	88%	> No drawing (2) > No text indicator (1)	No drawing (2)
C2 to G1	88%	92%	88%	> No drawing (2) > No text indicator (1)	No drawing (2)
C4 to G3	88%	92%	88%	> No drawing (2) > No text indicator (1)	No drawing (2)
C5 to G3	88%	92%	88%	> No drawing (2) > No text indicator (1)	No drawing (2)

TABLE D.14: Mistakes committed by Group A participants: SACM alternate notation AssertedInference

AssertedInference	% of participants who correctly drawn the element			Error found	
	Visual	Position	Both visual and position	Visual	Position
G3 to G1	88%	92%	88%	> No drawing (2) > No text indicator (1)	No drawing (2)
G7 to G3	83%	58%	58%	No drawing (4)	> No drawing (4) > Wrong place (6)
G8 to G3	83%	58%	58%	No drawing (4)	> No drawing (4) > Wrong place (6)

TABLE D.15: Mistakes committed by Group A participants: SACM alternate notation AssertedEvidence

AssertedEvidence	% of participants who correctly drawn the element			Error found	
	Visual	Position	Both visual and position	Visual	Position
Sn3 to G7	71%	83%	63%	> No drawing (1) > Wrong direction (1) > Drawn as an AssertedContext (1) > No line head (1) > No text indicator (3)	> No drawing (1) > Wrong place (3)
Sn4 to G8	71%	83%	63%	> No drawing (1) > Wrong direction (1) > Drawn as an AssertedContext (1) > No line head (1) > No text indicator (3)	> No drawing (1) > Wrong place (3)

D.4.3 Group B: SACM notation

This subsection presents mistakes committed by Group B participants in creating an assurance case model using SACM notation: Claim (Table D.16), ArgumentReasoning (Table D.17), ArtifactReference (Table D.18), AssertedContext (Table D.19), AssertedInference (Table D.20), and AssertedEvidence (Table D.21).

TABLE D.16: Mistakes committed by Group B participants: SACM notation Claim

Claim	% of participants who correctly drawn the element			Error found	
	Visual	Position	Both visual and position	Visual	Position
G1	91%	100%	91%	> Forgot VR (1) > Drawn as a rectangle (1) > Drawn as an ellipse (1)	N/A
G3	91%	89%	83%	> Forgot VR (1) > Drawn as an ellipse (1) > Drawn as an ArgumentReasoning (1)	Wrong place (4)
G7	86%	69%	57%	> Forgot VR (1) > Drawn as ArtifactReference (1) > Drawn as an ellipse (3)	Wrong place (11)
G8	86%	60%	54%	> Forgot VR (1) > No drawing (1) > Drawn as ArtifactReference (1) > Drawn as an ellipse (2)	Wrong place (13)

TABLE D.17: Mistakes committed by Group B participants: SACM notation ArgumentReasoning

ArgumentReasoning	% of participants who correctly drawn the element			Error found	
	Visual	Position	Both visual and position	Visual	Position
	66%	51%	49%	> Forgot VR (3) > No drawing (1) > Drawn as a rectangle (5) > Drawn as an ellipse (2)	> Wrong place (13) > No drawing (1)

TABLE D.18: Mistakes committed by Group B participants: SACM notation ArtifactReference

ArtifactReference	% of participants who correctly drawn the element			Error found	
	Visual	Position	Both visual and position	Visual	Position
C1	31%	83%	31%	> Forgot VR (2) > No arrow (12) > Drawn as a rectangle (8) > Drawn as an ellipse (2)	Wrong place (6)
C2	31%	83%	31%	> Forgot VR (2) > No arrow (12) > Drawn as a rectangle (8) > Drawn as an ellipse (2)	Wrong place (6)
C4	26%	71%	26%	> Forgot VR (2) > No drawing (1) > No arrow (13) > Drawn as a rectangle (8) > Drawn as an ellipse (2)	> Wrong place (9) > No drawing (1)
C5	23%	71%	23%	> Forgot VR (2) > No drawing (2) > No arrow (13) > Drawn as a rectangle (8) > Drawn as an ellipse (2)	Wrong place (8) No drawing (2)
Sn3	31%	77%	31%	> Forgot VR (3) > No arrow (10) > Drawn as a rectangle (9) > Drawn as an ellipse (2)	Wrong place (8)
Sn4	26%	80%	26%	> Forgot VR (3) > No arrow (12) > Drawn as a rectangle (9) > Drawn as an ellipse (2)	Wrong place (7)

TABLE D.19: Mistakes committed by Group B participants: SACM notation AssertedContext

AssertedContext	% of participants who correctly drawn the element			Error found	
	Visual	Position	Both visual and position	Visual	Position
C1 to G1	57%	83%	54%	> Wrong direction (7) > no dot (7) > no squarehead (5)	Wrong place (6)
C2 to G1	57%	83%	57%	> No drawing (1) > Wrong direction (6) > no dot (7) > no squarehead (5)	> No drawing (1) > Wrong place (5)
C4 to G3	51%	71%	49%	> No drawing (2) > Wrong direction (5) > no dot (8) > no squarehead (5)	> No drawing (2) > Wrong place (7)
C5 to G3	51%	71%	49%	> No drawing (3) > Wrong direction (5) > no dot (7) > no squarehead (5)	> No drawing (3) > Wrong place (6)

TABLE D.20: Mistakes committed by Group B participants: SACM notation AssertedInference

AssertedInference	% of participants who correctly drawn the element			Error found	
	Visual	Position	Both visual and position	Visual	Position
G3 to G1	57%	80%	54%	> No drawing (4) > No arrow (5) > Wrong direction (3) > no dot (3) > Wrong arrowhead (2) > no arrowhead (5)	> No drawing (4) > Wrong place (3)
G7 to G3	46%	51%	43%	> No drawing (16) > Wrong arrowhead (3)	> No drawing (16) > Wrong place (1)
G8 to G3	43%	51%	43%	> No drawing (17) > Wrong arrowhead (3)	No drawing (17)

TABLE D.21: Mistakes committed by Group B participants: SACM notation AssertedEvidence

AssertedEvidence	% of participants who correctly drawn the element			Error found	
	Visual	Position	Both visual and position	Visual	Position
Sn3 to G7	51%	83%	51%	> No arrow (8) > Wrong direction (3) > no dot (9) > Wrong arrowhead (6) > no arrowhead (8)	Wrong place (6)
Sn4 to G8	43%	80%	43%	> No drawing (2) > No arrow (9) > Wrong direction (3) > no dot (10) > Wrong arrowhead (6) > no arrowhead (9)	> No drawing (2) > Wrong place (5)

D.4.4 Group B: SACM alternate notation

This subsection presents mistakes committed by Group B participants in creating an assurance case model using SACM notation: Claim (Table D.22), ArtifactReference (Table D.23), ArgumentReasoning (Table D.24), AssertedContext (Table D.25), AssertedInference (Table D.26), and AssertedEvidence (Table D.27).

TABLE D.22: Mistakes committed by Group A participants: SACM alternate notation Claim

Claim	% of participants who correctly drawn the element			Error found	
	Visual	Position	Both visual and position	Visual	Position
G1	19%	100%	19%	> Not thick (17) > Not the biggest (18)	N/A
G3	100%	84%	84%	N/A	Wrong place (5)
G7	97%	78%	75%	Drawn as a Context (1)	Wrong place (7)
G8	97%	78%	75%	Drawn as a Context (1)	Wrong place (7)

TABLE D.23: Mistakes committed by Group A participants: SACM alternate notation ArtifactReference

ArtifactReference	% of participants who correctly drawn the element			Error found	
	Visual	Position	Both visual and position	Visual	Position
C1	41%	88%	38%	> Forgot VR (2) > No star (7) > Drawn as an ellipse (10)	Wrong place (4)
C2	41%	84%	38%	> Forgot VR (2) > No star (7) > Drawn as an ellipse (10)	Wrong place (5)
C4	34%	81%	34%	> Forgot VR (2) > No drawing (1) > No star (8) > Drawn as an ellipse (10)	> No drawing (1) > Wrong place (5)
C5	28%	78%	28%	> Forgot VR (2) > No drawing (1) > No star (10) > Drawn as an ellipse (10)	> No drawing (1) > Wrong place (6)
Sn3	44%	100%	44%	> Forgot VR (1) > No star (13) > Drawn as an ellipse (4)	N/A
Sn4	34%	100%	34%	> Forgot VR (1) > No star (16) > Drawn as an ellipse (4)	N/A

TABLE D.24: Mistakes committed by Group A participants: SACM alternate notation ArgumentReasoning

ArgumentReasoning	% of participants who correctly drawn the element			Error found	
	Visual	Position	Both visual and position	Visual	Position
	66%	72%	53%	> Forgot VR (3) > No drawing (2) > Drawn as an ellipse (6)	> No drawing (2) > Wrong place (7)

TABLE D.25: Mistakes committed by Group A participants: SACM alternate notation AssertedContext

AssertedContext	% of participants who correctly drawn the element			Error found	
	Visual	Position	Both visual and position	Visual	Position
C1 to G1	28%	88%	28%	> No text indicator (14) > Drawn as a single line (7) > Drawn as AssertedEvidence (2)	Wrong place (4)
C2 to G1	28%	84%	28%	> No drawing (1) > No text indicator (14) > Drawn as a single line (6) > Drawn as AssertedEvidence (2)	> No drawing (1) > Wrong place (4)
C4 to G3	22%	78%	22%	> No drawing (1) > No text indicator (14) > Drawn as a single line (9) > Drawn as AssertedEvidence (1)	> No drawing (1) > Wrong place (6)
C5 to G3	22%	78%	22%	> No drawing (2) > No text indicator (14) > Drawn as a single line (8) > Drawn as AssertedEvidence (1)	> No drawing (2) > Wrong place (5)

TABLE D.26: Mistakes committed by Group A participants: SACM alternate notation AssertedInference

AssertedInference	% of participants who correctly drawn the element			Error found	
	Visual	Position	Both visual and position	Visual	Position
G3 to G1	53%	81%	53%	> No drawing (4) > Drawn as an AssertedContext (1) > No text indicator (8) > Drawn as AssertedEvidence (2)	> No drawing (4) > Wrong place (2)
G7 to G3	56%	72%	44%	> No drawing (3) > No text indicator (9) > Drawn as AssertedEvidence (2)	> No drawing (3) > Wrong place (6)
G8 to G3	56%	75%	44%	> No drawing (3) > No text indicator (9) > Drawn as AssertedEvidence (2)	> No drawing (3) > Wrong place (5)

TABLE D.27: Mistakes committed by Group A participants: SACM alternate notation AssertedEvidence

AssertedEvidence	% of participants who correctly drawn the element			Error found	
	Visual	Position	Both visual and position	Visual	Position
Sn3 to G7	41%	100%	41%	> No line head (1) > No text indicator (15) > Drawn as an AssertedInference (4)	N/A
Sn4 to G8	34%	100%	34%	> No line head (1) > No text indicator (17) > Drawn as an AssertedInference (4)	N/A

D.5 Identification of element variants result

This section presents the result of the identification of element variants of SACM notation and SACM alternate notation.

D.5.1 SACM notation elements

This subsection presents the result of the identification of element variants of SACM notation.

TABLE D.28: Participants' responses related to the identification of 'Claim' variants (in SACM notation vs Alt. SACM notation study)

Info given Question	Claim					
	Identify Claim variants					
Answer option	Group A		Group B		Group A and B	
	22 participants		32 participants		Total 54 participants	
	Count	%	Count	%	Count	%
<i>ArtifactReference</i>	3	14%	6	19%	9	17%
<i>ArgumentPackage</i>	3	14%	5	16%	8	15%
<i>AssumedClaim</i>	12	55%	19	59%	31	57%
<i>AsCitedClaim</i>	13	59%	16	50%	29	54%
<i>AxiomaticClaim</i>	14	64%	22	69%	36	67%
<i>DefeatedClaim</i>	3	14%	14	44%	17	31%
<i>NeedsSupportClaim</i>	9	41%	22	69%	31	57%
<i>AbstractClaim</i>	9	41%	19	59%	28	52%
<i>ArgumentReasoning</i>	1	5%	3	9%	4	7%
<i>AssertedInference</i>	4	18%	2	6%	6	11%
<i>AssumedContext</i>	1	5%	0	0%	1	2%
<i>AxiomaticInference</i>	1	5%	0	0%	1	2%
<i>AssumedInference</i>	1	5%	0	0%	1	2%
<i>CounterInference</i>	0	0%	1	3%	1	2%
<i>MetaClaim</i>	0	0%	0	0%	0	0%
<i>ArgPkgBinding</i>	2	9%	4	13%	6	11%
<i>ArgPkgInterface</i>	2	9%	4	13%	6	11%

TABLE D.29: Participants' responses related to the identification of 'ArgumentPackage' variants (in SACM notation vs Alt. SACM notation study)

Info given Question	ArgumentPackage					
	Identify ArgumentPackage variants					
Answer option	Group A		Group B		Group A and B	
	22 participants		32 participants		Total 54 participants	
	Count	%	Count	%	Count	%
<i>ArtifactReference</i>	4	18%	1	3%	5	9%
<i>Claim</i>	1	5%	2	6%	3	6%
<i>AssumedClaim</i>	1	5%	0	0%	1	2%
<i>AsCitedClaim</i>	3	14%	2	6%	5	9%
<i>AxiomaticClaim</i>	0	0%	0	0%	0	0%
<i>DefeatedClaim</i>	0	0%	0	0%	0	0%
<i>NeedsSupportClaim</i>	1	5%	0	0%	1	2%
<i>AbstractClaim</i>	1	5%	0	0%	1	2%
<i>ArgumentReasoning</i>	2	9%	2	6%	4	7%
<i>AssertedInference</i>	1	5%	1	3%	2	4%
<i>AssumedContext</i>	1	5%	0	0%	1	2%
<i>AxiomaticInference</i>	0	0%	0	0%	0	0%
<i>AssumedInference</i>	0	0%	0	0%	0	0%
<i>CounterInference</i>	0	0%	0	0%	0	0%
<i>MetaClaim</i>	0	0%	0	0%	0	0%
<i>ArgPkgBinding</i>	17	77%	29	91%	46	85%
<i>ArgPkgInterface</i>	17	77%	24	75%	41	76%

TABLE D.30: Participants' responses related to the identification of 'assumed' element variants (in SACM notation vs Alt. SACM notation study)

Info given	AssumedClaim					
Question	Identify other assumed elements					
Answer option	Group A		Group B		Group A and B	
	22 participants		32 participants		Total 54 participants	
	Count	%	Count	%	Count	%
<i>ArtifactReference</i>	1	5%	1	3%	2	4%
<i>Claim</i>	0	0%	2	6%	2	4%
<i>AsCitedClaim</i>	5	23%	8	25%	13	24%
<i>AxiomaticClaim</i>	4	18%	3	9%	7	13%
<i>DefeatedClaim</i>	3	14%	1	3%	4	7%
<i>NeedsSupportClaim</i>	16	73%	14	44%	30	56%
<i>AbstractClaim</i>	7	32%	4	13%	11	20%
<i>ArgumentReasoning</i>	1	5%	1	3%	2	4%
<i>AssertedInference</i>	0	0%	2	6%	2	4%
<i>AssumedContext</i>	1	5%	14	44%	15	28%
<i>AxiomaticInference</i>	1	5%	5	16%	6	11%
<i>AssumedInference</i>	1	5%	16	50%	17	31%
<i>CounterInference</i>	0	0%	1	3%	1	2%
<i>MetaClaim</i>	1	5%	0	0%	1	2%
<i>ArgPkgBinding</i>	0	0%	0	0%	0	0%
<i>ArgPkgInterface</i>	0	0%	0	0%	0	0%

TABLE D.31: Participants' responses related to the identification of 'axiomatic' element variants (in SACM notation vs Alt. SACM notation study)

Info given	AxiomaticInference					
Question	Identify other Axiomatic elements					
Answer option	Group A		Group B		Group A and B	
	22 participants		32 participants		Total 54 participants	
	Count	%	Count	%	Count	%
<i>ArtifactReference</i>	1	5%	0	0%	1	2%
<i>Claim</i>	1	5%	0	0%	1	2%
<i>AsCitedClaim</i>	0	0%	0	0%	0	0%
<i>AxiomaticClaim</i>	1	5%	10	31%	11	20%
<i>DefeatedClaim</i>	1	5%	0	0%	1	2%
<i>NeedsSupportClaim</i>	0	0%	1	3%	1	2%
<i>AbstractClaim</i>	0	0%	0	0%	0	0%
<i>ArgumentReasoning</i>	1	5%	1	3%	2	4%
<i>AssertedInference</i>	4	18%	11	34%	15	28%
<i>AssumedContext</i>	5	23%	4	13%	9	17%
<i>AxiomaticContext</i>	12	55%	23	72%	35	65%
<i>AssumedInference</i>	10	45%	8	25%	18	33%
<i>CounterInference</i>	2	9%	10	31%	12	22%
<i>MetaClaim</i>	3	14%	4	13%	7	13%
<i>ArgPkgBinding</i>	0	0%	0	0%	0	0%
<i>ArgPkgInterface</i>	0	0%	0	0%	0	0%

TABLE D.32: Participants' responses related to the identification of 'defeated' element variants (in SACM notation vs Alt. SACM notation study)

Info given	DefeatedClaim					
Question	Identify other Defeated elements					
Answer option	Group A		Group B		Group A and B	
	22 participants		32 participants		Total 54 participants	
	Count	%	Count	%	Count	%
<i>ArtifactReference</i>	0	0%	0	0%	0	0%
<i>Claim</i>	0	0%	2	6%	2	4%
<i>AsCitedClaim</i>	1	5%	0	0%	1	2%
<i>AxiomaticClaim</i>	0	0%	2	6%	2	4%
<i>DefeatedInference</i>	12	55%	28	88%	40	74%
<i>NeedsSupportClaim</i>	0	0%	1	3%	1	2%
<i>AbstractClaim</i>	2	9%	3	9%	5	9%
<i>ArgumentReasoning</i>	0	0%	0	0%	0	0%
<i>AssertedInference</i>	0	0%	0	0%	0	0%
<i>DefeatedContext</i>	7	32%	27	84%	34	63%
<i>AxiomaticContext</i>	0	0%	0	0%	0	0%
<i>AssumedInference</i>	0	0%	0	0%	0	0%
<i>CounterInference</i>	0	0%	0	0%	0	0%
<i>MetaClaim</i>	0	0%	0	0%	0	0%
<i>ArgPkgBinding</i>	0	0%	0	0%	0	0%
<i>ArgPkgInterface</i>	0	0%	0	0%	0	0%

TABLE D.33: Participants' responses related to the identification of 'abstract' element variants (in SACM notation vs Alt. SACM notation study)

Info given	AbstractClaim					
Question	Identify other Abstract elements					
Answer option	Group A		Group B		Group A and B	
	22 participants		32 participants		Total 54 participants	
	Count	%	Count	%	Count	%
<i>AbstractArtifactReference</i>	18	82%	27	84%	45	83%
<i>Claim</i>	1	5%	0	0%	1	2%
<i>AsCitedClaim</i>	1	5%	3	9%	4	7%
<i>AxiomaticClaim</i>	0	0%	1	3%	1	2%
<i>DefeatedInference</i>	0	0%	0	0%	0	0%
<i>NeedsSupportClaim</i>	1	5%	2	6%	3	6%
<i>AbstractInference</i>	11	50%	28	88%	39	72%
<i>ArgumentReasoning</i>	0	0%	0	0%	0	0%
<i>AssertedInference</i>	0	0%	0	0%	0	0%
<i>DefeatedContext</i>	0	0%	0	0%	0	0%
<i>AbstractContext</i>	10	45%	24	75%	34	63%
<i>AssumedInference</i>	0	0%	0	0%	0	0%
<i>CounterInference</i>	0	0%	0	0%	0	0%
<i>MetaClaim</i>	0	0%	0	0%	0	0%
<i>ArgPkgBinding</i>	0	0%	0	0%	0	0%
<i>ArgPkgInterface</i>	0	0%	0	0%	0	0%

TABLE D.34: Participants' responses related to the identification of 'asCited' element variants (in SACM notation vs Alt. SACM notation study)

Info given	AsCitedClaim					
Question	Identify other AsCited elements					
Answer option	Group A		Group B		Group A and B	
	22 participants		32 participants		Total 54 participants	
	Count	%	Count	%	Count	%
<i>AbstractArtifactReference</i>	2	9%	2	6%	4	7%
<i>Claim</i>	0	0%	0	0%	0	0%
<i>AsCitedInference</i>	12	55%	20	63%	32	59%
<i>AxiomaticClaim</i>	3	14%	5	16%	8	15%
<i>DefeatedInference</i>	0	0%	0	0%	0	0%
<i>NeedsSupportClaim</i>	1	5%	3	9%	4	7%
<i>AbstractInference</i>	0	0%	0	0%	0	0%
<i>ArgumentReasoning</i>	0	0%	5	16%	5	9%
<i>AssertedInference</i>	0	0%	0	0%	0	0%
<i>DefeatedContext</i>	1	5%	0	0%	1	2%
<i>AsCitedContext</i>	9	41%	24	75%	33	61%
<i>AssumedInference</i>	2	9%	1	3%	3	6%
<i>CounterInference</i>	0	0%	0	0%	0	0%
<i>MetaClaim</i>	0	0%	0	0%	0	0%
<i>ArgPkgBinding</i>	0	0%	6	19%	6	11%
<i>ArgPkgInterface</i>	0	0%	2	6%	2	4%

TABLE D.35: Participants' responses related to the identification of 'NeedsSupport' element variants (in SACM notation vs Alt. SACM notation study)

Info given	NeedsSupportInference					
Question	Identify other AsCited elements					
Answer option	Group A		Group B		Group A and B	
	22 participants		32 participants		Total 54 participants	
	Count	%	Count	%	Count	%
<i>AbstractArtifactReference</i>	0	0%	0	0%	0	0%
<i>Claim</i>	0	0%	0	0%	0	0%
<i>AsCitedInference</i>	2	9%	3	9%	5	9%
<i>AxiomaticClaim</i>	0	0%	1	3%	1	2%
<i>DefeatedInference</i>	0	0%	2	6%	2	4%
<i>NeedsSupportClaim</i>	10	45%	24	75%	34	63%
<i>AbstractInference</i>	2	9%	1	3%	3	6%
<i>ArgumentReasoning</i>	0	0%	0	0%	0	0%
<i>AssertedInference</i>	0	0%	3	9%	3	6%
<i>DefeatedContext</i>	0	0%	1	3%	1	2%
<i>NeedSupportContext</i>	19	86%	26	81%	45	83%
<i>AssumedInference</i>	4	18%	3	9%	7	13%
<i>CounterInference</i>	0	0%	3	9%	3	6%
<i>MetaClaim</i>	1	5%	0	0%	1	2%
<i>ArgPkgBinding</i>	0	0%	0	0%	0	0%
<i>ArgPkgInterface</i>	0	0%	2	6%	2	4%

D.5.2 Alternate SACM notation elements

This subsection presents the result of the identification of element variants of SACM alternate notation.

TABLE D.36: Participants' responses related to the identification of 'Claim' variants of the Alt. SACM notation

Info given	AltClaim					
Question	Identify Claim variants					
Answer option	Group A		Group B		Group A and B	
	22 participants		32 participants		Total 54 participants	
	Count	%	Count	%	Count	%
<i>AltAxiomaticClaim</i>	16	73%	26	81%	42	78%
<i>AltArgumentPackage</i>	0	0%	2	6%	2	4%
<i>AltAssumedClaim</i>	14	64%	20	63%	34	63%
<i>AltAbstractClaim</i>	16	73%	24	75%	40	74%
<i>AltDefeatedClaim</i>	12	55%	21	66%	33	61%
<i>AltNeedsSupportClaim</i>	14	64%	19	59%	33	61%
<i>AltAsCitedClaim</i>	14	64%	21	66%	35	65%
<i>AltMetaClaim</i>	1	5%	4	13%	5	9%
<i>AltAssumedEvidence</i>	0	0%	2	6%	2	4%
<i>AltAssumedInference</i>	1	5%	0	0%	1	2%
<i>AltAbstractContext</i>	0	0%	1	3%	1	2%

TABLE D.37: Participants' responses related to the identification of 'assumed' element variants of the Alt. SACM notation

Info given	AltAssumedClaim					
Question	Identify other assumed elements					
Answer option	Group A		Group B		Group A and B	
	22 participants		32 participants		Total 54 participants	
	Count	%	Count	%	Count	%
<i>AltArtifactReference</i>	2	9%	2	6%	4	7%
<i>AltAssertedClaim</i>	1	5%	6	19%	7	13%
<i>AltAsCitedClaim</i>	2	9%	6	19%	8	15%
<i>AltAxiomaticClaim</i>	3	14%	9	28%	12	22%
<i>AltDefeatedClaim</i>	2	9%	7	22%	9	17%
<i>AltNeedsSupportClaim</i>	8	36%	12	38%	20	37%
<i>AltAbstractClaim</i>	3	14%	4	13%	7	13%
<i>AltArgumentReasoning</i>	0	0%	0	0%	0	0%
<i>AltAssertedInference</i>	0	0%	1	3%	1	2%
<i>AltAssumedContext</i>	10	45%	14	44%	24	44%
<i>AltAxiomaticInference</i>	0	0%	1	3%	1	2%
<i>AltAssumedInference</i>	10	45%	17	53%	27	50%
<i>AltAssumedEvidence</i>	11	50%	12	38%	23	43%
<i>AltMetaClaim</i>	1	5%	3	9%	4	7%
<i>AltArgumentPackageBinding</i>	0	0%	0	0%	0	0%
<i>AltArgumentPackageInterface</i>	0	0%	1	3%	1	2%

TABLE D.38: Participants' responses related to the identification of 'ArgumentPackage' variants of the Alt. SACM notation

Info given	AltArgumentPackage					
Question	Identify ArgumentPackage variants					
Answer option	Group A		Group B		Group A and B	
	22 participants		32 participants		Total 54 participants	
	Count	%	Count	%	Count	%
<i>AltAxiomaticClaim</i>	1	5%	0	0%	1	2%
<i>AltArgumentPackageBinding</i>	17	77%	24	75%	41	76%
<i>AltAssumedClaim</i>	1	5%	1	3%	2	4%
<i>AltAbstractClaim</i>	0	0%	2	6%	2	4%
<i>AltArgumentPackageInterface</i>	18	82%	31	97%	49	91%
<i>AltAssumedEvidence</i>	2	9%	0	0%	2	4%
<i>AltAssumedInference</i>	2	9%	1	3%	3	6%

TABLE D.39: Participants' responses related to the identification of 'axiomatic' element variants of the Alt. SACM notation

Info given	AltAxiomaticInference					
Question	Identify other Axiomatic elements					
Answer option	Group A		Group B		Group A and B	
	22 participants		32 participants		Total 54 participants	
	Count	%	Count	%	Count	%
<i>AltArtifactReference</i>	1	5%	2	6%	3	6%
<i>AltAssertedClaim</i>	1	5%	0	0%	1	2%
<i>AltAsCitedClaim</i>	0	0%	2	6%	2	4%
<i>AltAxiomaticClaim</i>	9	41%	12	38%	21	39%
<i>AltDefeatedClaim</i>	0	0%	0	0%	0	0%
<i>AltNeedsSupportClaim</i>	0	0%	0	0%	0	0%
<i>AltAbstractClaim</i>	0	0%	0	0%	0	0%
<i>AltArgumentReasoning</i>	0	0%	1	3%	1	2%
<i>AltAssertedInference</i>	4	18%	7	22%	11	20%
<i>AltAssumedContext</i>	4	18%	7	22%	11	20%
<i>AltAxiomaticContext</i>	16	73%	26	81%	42	78%
<i>AltAxiomaticEvidence</i>	2	9%	26	81%	28	52%
<i>AltCounterInference</i>	2	9%	6	19%	8	15%
<i>AltMetaClaim</i>	1	5%	6	19%	7	13%
<i>AltArgumentPackageBinding</i>	0	0%	0	0%	0	0%
<i>AltArgumentPackageInterface</i>	0	0%	0	0%	0	0%

TABLE D.40: Participants' responses related to the identification of 'defeated' element variants of the Alt. SACM notation

Info given	AltDefeatedClaim					
Question	Identify other Defeated elements					
Answer option	Group A		Group B		Group A and B	
	22 participants		32 participants		Total 54 participants	
	Count	%	Count	%	Count	%
<i>AltArtifactReference</i>	0	0%	2	6%	2	4%
<i>AltAssertedClaim</i>	0	0%	2	6%	2	4%
<i>AltAsCitedClaim</i>	0	0%	3	9%	3	6%
<i>AltAxiomaticClaim</i>	0	0%	4	13%	4	7%
<i>AltDefeatedInference</i>	14	64%	21	66%	35	65%
<i>AltNeedsSupportClaim</i>	0	0%	4	13%	4	7%
<i>AltAbstractClaim</i>	2	9%	6	19%	8	15%
<i>AltArgumentReasoning</i>	0	0%	1	3%	1	2%
<i>AltAssertedInference</i>	0	0%	1	3%	1	2%
<i>AltAssumedContext</i>	0	0%	0	0%	0	0%
<i>AltDefeatedEvidence</i>	16	73%	19	59%	35	65%
<i>AltAssumedInference</i>	0	0%	1	3%	1	2%
<i>AltDefeatedContext</i>	17	77%	18	56%	35	65%
<i>AltMetaClaim</i>	0	0%	1	3%	1	2%
<i>AltArgumentPackageBinding</i>	1	5%	0	0%	1	2%
<i>AltArgumentPackageInterface</i>	1	5%	0	0%	1	2%

TABLE D.41: Participants' responses related to the identification of 'abstract' element variants of the Alt. SACM notation

Info given	AltAbstractClaim					
Question	Identify other Abstract elements					
Answer option	Group A		Group B		Group A and B	
	22 participants		32 participants		Total 54 participants	
	Count	%	Count	%	Count	%
<i>AltArtifactReference</i>	0	0%	1	3%	1	2%
<i>AltAssertedClaim</i>	1	5%	5	16%	6	11%
<i>AltAsCitedClaim</i>	1	5%	7	22%	8	15%
<i>AltAxiomaticClaim</i>	1	5%	0	0%	1	2%
<i>AltDefeatedInference</i>	1	5%	0	0%	1	2%
<i>AltNeedsSupportClaim</i>	1	5%	6	19%	7	13%
<i>AltAbstractInference</i>	16	73%	25	78%	41	76%
<i>AltArgumentReasoning</i>	0	0%	0	0%	0	0%
<i>AltAssertedInference</i>	0	0%	2	6%	2	4%
<i>AltAbstractEvidence</i>	15	68%	21	66%	36	67%
<i>AltDefeatedEvidence</i>	0	0%	0	0%	0	0%
<i>AltAbstractContext</i>	16	73%	20	63%	36	67%
<i>AltDefeatedContext</i>	1	5%	0	0%	1	2%
<i>AltMetaClaim</i>	0	0%	0	0%	0	0%
<i>AltArgumentPackageBinding</i>	0	0%	1	3%	1	2%
<i>AltArgumentPackageInterface</i>	1	5%	2	6%	3	6%

TABLE D.42: Participants' responses related to the identification of 'asCited' element variants of the Alt. SACM notation

Info given	AltAsCitedClaim					
Question	Identify other AsCited elements					
Answer option	Group A		Group B		Group A and B	
	22 participants		32 participants		Total 54 participants	
	Count	%	Count	%	Count	%
<i>AltArtifactReference</i>	0	0%	4	13%	4	7%
<i>AltAssertedClaim</i>	0	0%	2	6%	2	4%
<i>AltAsCitedInference</i>	15	68%	25	78%	40	74%
<i>AltAxiomaticClaim</i>	0	0%	6	19%	6	11%
<i>AltDefeatedInference</i>	0	0%	0	0%	0	0%
<i>AltNeedsSupportClaim</i>	2	9%	0	0%	2	4%
<i>AltAbstractInference</i>	0	0%	0	0%	0	0%
<i>AltArgumentReasoning</i>	1	5%	1	3%	2	4%
<i>AltAssertedInference</i>	1	5%	0	0%	1	2%
<i>AltAbstractEvidence</i>	0	0%	0	0%	0	0%
<i>AltAsCitedEvidence</i>	14	64%	25	78%	39	72%
<i>AltAbstractContext</i>	0	0%	0	0%	0	0%
<i>AltAsCitedContext</i>	15	68%	25	78%	40	74%
<i>AltMetaClaim</i>	0	0%	0	0%	0	0%
<i>AltArgumentPackageBinding</i>	0	0%	2	6%	2	4%
<i>AltArgumentPackageInterface</i>	0	0%	0	0%	0	0%

TABLE D.43: Participants' responses related to the identification of 'needsSupport' element variants of the Alt. SACM notation

Info given	AltNeedsSupportInference					
Question	Identify other NeedsSupport elements					
Answer option	Group A		Group B		Group A and B	
	22 participants		32 participants		Total 54 participants	
	Count	%	Count	%	Count	%
<i>AltArtifactReference</i>	0	0%	0	0%	0	0%
<i>AltAssertedClaim</i>	0	0%	2	6%	2	4%
<i>AltAsCitedClaim</i>	2	9%	1	3%	3	6%
<i>AltAxiomaticClaim</i>	1	5%	0	0%	1	2%
<i>AltDefeatedClaim</i>	0	0%	1	3%	1	2%
<i>AltNeedsSupportClaim</i>	14	64%	22	69%	36	67%
<i>AltAbstractClaim</i>	0	0%	1	3%	1	2%
<i>AltArgumentReasoning</i>	1	5%	1	3%	2	4%
<i>AltNeedsSupportContext</i>	18	82%	26	81%	44	81%
<i>AltAssumedContext</i>	2	9%	3	9%	5	9%
<i>AltAxiomaticContext</i>	2	9%	3	9%	5	9%
<i>AltNeedsSupportEvidence</i>	18	82%	28	88%	46	85%
<i>AltCounterInference</i>	0	0%	2	6%	2	4%
<i>AltMetaClaim</i>	0	0%	0	0%	0	0%
<i>AltArgumentPackageBinding</i>	0	0%	2	6%	2	4%
<i>AltArgumentPackageInterface</i>	0	0%	0	0%	0	0%

D.6 Visual variables utilisation result

Layout and arrow direction

There were four questions asked to the participants related to the utilisation of visual variable Shape to represents relationship elements.

The first question is illustrated in Figure D.2. The participants were asked to choose which Claim that they think as the conclusion Claim.

If there are two claims in a diagram (Claim A and B), one as a premise and another is the conclusion.
According to the following diagram, in your opinion, which Claim indicates the conclusion claim?

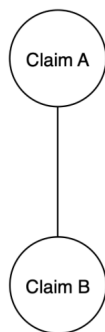


FIGURE D.2: Role of arrow direction in representing relationship element: Question #1

Table D.44 presents the participants' responses related to the first question.

TABLE D.44: Participants' response regarding the first question related to the role of the arrowhead to represent relationship element

Answer option	A (22)		B (32)		Total A+B	
	Count	%	Count	%	Count	%
Claim A	8	36%	5	16%	13	24%
Claim B	3	14%	2	6%	5	9%
None of them	10	45%	8	25%	18	33%
Both of them	1	5%	17	53%	18	33%
No answer	0	0%	0	0%	0	0%

As for the second question is illustrated in Figure D.3. The participants were asked which Claim is the conclusion Claim. Table D.45 presents the participants responses regarding the second question.

If there are two claims in a diagram (Claim A and B), one as a premise and another is the conclusion.
According to the following diagram, in your opinion, which Claim indicates the conclusion claim?



FIGURE D.3: Role of arrow direction in representing relationship element: Question #2

TABLE D.45: Participants' response regarding the second question related to the role of the arrowhead to represent relationship element (in SACM notation vs Alt. SACM notation study)

Answer option	A (22)		B (32)		Total A+B	
	Count	%	Count	%	Count	%
Claim A	17	77%	21	66%	38	70%
Claim B	3	14%	7	22%	10	19%
None of them	2	9%	4	13%	6	11%
Both of them	0	0%	0	0%	0	0%
No answer	0	0%	0	0%	0	0%

The third question is illustrated in Figure D.4. The participants' responses regarding the third question is presented in Table D.46.

If there are two claims in a diagram (Claim A and B), one as a premise and another is the conclusion.
According to the following diagram, in your opinion, which Claim indicates the conclusion claim?



FIGURE D.4: Role of arrow direction in representing relationship element: Question #3

TABLE D.46: Participants' response regarding the third question related to the role of the arrowhead to represent relationship element (in SACM notation vs Alt. SACM notation study)

Answer option	A (22)		B (32)		Total A+B	
	Count	%	Count	%	Count	%
Claim A	1	5%	6	19%	7	13%
Claim B	2	9%	3	9%	5	9%
None of them	15	68%	10	31%	25	46%
Both of them	4	18%	13	41%	17	31
No answer	0	0%	0	0%	0	0%

The fourth question is illustrated in Figure D.5. The participants were asked to choose which Claim they think as the conclusion Claim. The participants' responses regarding this question is presented in Table D.47.

If there are two claims in a diagram (Claim A and B), one as a premise and another is the conclusion.
According to the following diagram, in your opinion, which Claim indicates the conclusion claim?



FIGURE D.5: Role of arrow direction in representing relationship element: Question #4

TABLE D.47: Participants' response regarding the fourth question related to the role of the arrowhead to represent relationship element

Answer option	A (22)		B (32)		Total A+B	
	Count	%	Count	%	Count	%
Claim A	7	32%	6	19%	13	24%
Claim B	15	68%	21	66%	36	67%
None of them	0	0%	4	13%	4	7%
Both of them	0	0%	1	3%	1	2%
No answer	0	0%	0	0%	0	0%

Size in the context of line thickness

The participants were also asked about the utilisation of Size visual variable in the context of the line size thickness as used to represent the 'axiomatic' element in SACM notation. In this case, the participants were provided with a diagram of Claim A and B located horizontally relative to each other positions. Both Claims were represented using a circle and connected using a line as without any decoration to indicate the source or target element. The circle that represents Claim A was drawn thicker than Claim B (as illustrated in Figure D.6). The responses from the participants regarding this question is presented in Table D.48.

If there are two claims (Claim A and B) in a diagram that are rendered with different emphasise of the size of line thickness, in your opinion, which Claim indicates the conclusion claim?



FIGURE D.6: Role of line thickness in notation design

TABLE D.48: Participants' responses regarding the question related to the role of visual variable Size in the context of line thickness (in SACM notation vs Alt. SACM notation study)

Answer option	A (22)		B (32)		Total A+B	
	Count	%	Count	%	Count	%
Claim A	17	77%	18	56%	35	65%
Claim B	3	14%	12	38%	15	28%
None of them	1	5%	2	6%	3	6%
Both of them	1	5%	0	0%	1	2%
No answer	0	0%	0	0%	0	0%

Role of Brightness

The last question was designed to investigate the participants' perception regarding the use of a particular value of Brightness (visual variable) in a notation design (as illustrated in Figure D.7).

Please take a look at the following diagram.



If the dark (brightness) circle-head line indicates a relationship that Claim A is supported by Claim B, in your opinion, what can be indicated by a light (brightness) circle-head line used as a relationship between Claim A and B as shown in the following diagram?



FIGURE D.7: Role of Brightness in notation design

Table D.49 present the participants' responses regarding this question.

TABLE D.49: Participants' responses regarding the role of Brightness in the context of relationship visual representation design (in SACM notation vs Alt. SACM notation study)

Answer option	A (22)		B (32)		Total A+B	
	Count	%	Count	%	Count	%
Claim A is supported by Claim B	3	14%	4	13%	7	13%
Claim A is opposed by Claim B	11	50%	14	44%	25	46%
Claim A provide context to Claim B	3	14%	0	0%	3	6%
Claim B provide context to Claim A	5	23%	10	31%	15	28%
Nothing can be inferred	0	0%	3	9%	3	6%
Other	0	0%	1	3%	1	2%
No answer	0	0%	0	0%	0	0%

The answer that we are expected from the participants was: "Claim A is opposed by Claim B". This is due to help us to confirm that if the dark brightness used to indicates "support" relationship, the light brightness could be used (intuitively) to indicates the opposite of "support". Based on the result as presented in Table D.49, most of the participants from both groups agreed that the light brightness circle in this context provide meaning as the opposite of the support relationship. This result shows support to our design decision regarding the 'Counter' relationship in SACM notation, in this case, the 'Counter' relationship was visually represented using dark brightness such as in dark brightness arrowhead line to represent the 'Counter AssertedEvidence'.

D.7 Participants' perception towards the notations

TABLE D.50: Participants' perception regarding the notation that is relatively easy to learn

Q:	Which notation do you considered as the notation that is relatively easy to learn?	Group A		Group B		Group A & B	
		22 participants		32 participants		54 participants	
		Count	%	Count	%	Total	%
A:	SACM notation	10	45%	15	47%	25	46%
	Alt. SACM notation	6	27%	10	31%	16	30%
	None of them	1	5%	3	9%	4	7%
	Both of them	5	23%	3	9%	8	15%
	No answer	0	0%	1	3%	1	2%

TABLE D.51: Participants' perception regarding the notation that is relatively easy to use

Q:	Which notation do you considered as the notation that is relatively easy to use?	Group A		Group B		Group A & B	
		22 participants		32 participants		54 participants	
		Count	%	Count	%	Total	%
A:	SACM notation	15	68%	11	34%	26	48%
	Alt. SACM notation	7	32%	14	44%	21	39%
	None of them	0	0%	4	13%	4	7%
	Both of them	0	0%	2	6%	2	4%
	No answer	0	0%	1	3%	1	2%

TABLE D.52: Participants' perception regarding the notation that they intend to use in the future

Q:	Which notation do you intend to use in practice when being asked to construct an assurance case?	Group A		Group B		Group A & B	
		22 participants		32 participants		54 participants	
		Count	%	Count	%	Total	%
A:	SACM notation	10	45%	20	63%	30	56%
	Alt. SACM notation	7	32%	8	25%	15	28%
	None of them	0	0%	1	3%	1	2%
	Both of them	5	23%	3	9%	8	15%
	No answer	0	0%	0	0%	0	0%

D.8 Experienced user survey feedback

Table D.53 presents participants' feedback (experienced user survey) regarding the visual representation of the SACM elements

TABLE D.53: Participants' feedback regarding the visual representation of the SACM elements

Element	Visual representation	
	Advantages	Disadvantages
MetaClaim	<ul style="list-style-type: none"> >"Can express confidence in the Assertion (because there is no specific methodology for this in GSN), a good VR." >"Easy to distinguish from the other connector types." >"Good VR (this is very simple and very effective. Something I could easily explain to the clients/stakeholders.)" 	<ul style="list-style-type: none"> >"Disadvantage is that it clutters up the argument structure, making it difficult to follow and expanding the size when something smaller would do." >"The notation is subtle and may not be distinguishable from other link types if the scaling of presentation is not appropriate (e.g. if its too 'zoomed out')" >"It is not clear why three feet in the connector should reflect a meta-claim. This makes me think more of cardinalities in connections like in UML/SysML."
Counter Inference	<ul style="list-style-type: none"> >"The symbol and its use appears straight forward and clear." 	<ul style="list-style-type: none"> >"It's not very visually different to the other (relationship) elements." >"The open arrow head may be misinterpreted by those familiar with GSN (inContextRelationship)." >"The connector is not visibly different enough."
Counter Evidence	<ul style="list-style-type: none"> >"I think its good and like the way it is based on the counter inference notation being 're-purposed'." 	<ul style="list-style-type: none"> >"Shapes/links being very similar in the SACM visualisation." >"This is definitely going to confuse those with a GSN background. It's negative connotation is not intuitively clear from the symbol." >"The visual representation is the same as 'counter inference'. If you want to differentiate between the two then chose a different representation."
Defeated Claim	<ul style="list-style-type: none"> >"Good. Its obvious what it means." >"It would clearly identify weaknesses in the assurance case." >"This could provide a historic overview when explaining how the final SACM model has been derived. This is vitally important for complex projects or where there is an extended lifecycle." 	<ul style="list-style-type: none"> >"Could mistake it for being deleted rather than wrong/defeated."
ArgumentPackageInterface	<ul style="list-style-type: none"> >"Seems very intuitive to me, I have no problems with it." >"Advantages are the 'binding' packages are easily identifiable." 	<ul style="list-style-type: none"> >"I'd make the ArgumentPackageBinding more obviously different." >"Symbols are not intuitive (the 'link' symbol is OK, but the interface is not. Perhaps its more familiar to those that use UML?)" >"The notation looks overly complicated and is perhaps trying to say too much." >"Disadvantages are the 'binding' package could point to the packages it is binding."
ArgumentGroup	<ul style="list-style-type: none"> >"Even better if they were in colour. Biggest concern would be overlapping dotted lines." >"No disadvantages immediately come to mind." 	<ul style="list-style-type: none"> >"May be difficult if the grouping doesn't join up as a single blob." >"Difficulties may be experienced where elements of an argument aren't conveniently co-located and boundaries cross." >"Looks scruffy, does not scale to more than two or three groups."

References

- [1] Tim Kelly and Rob Weaver. The goal structuring notation—a safety argument notation. In *Proceedings of the dependable systems and networks 2004 workshop on assurance cases*, page 6. Citeseer, 2004.
- [2] Metamodel of the basic notation for toska. <http://www.vino4tosca.org>. Accessed: 2020-03-05.
- [3] D. Moody. The physics of notations: Toward a scientific basis for constructing visual notations in software engineering. *IEEE Transactions on Software Engineering*, 35(6):756–779, 2009. doi: 10.1109/TSE.2009.67.
- [4] Bertin Jacques. *Semiology of graphics: diagrams, networks, maps*. University of Wisconsin Press, Madison, Wisconsin, 1983.
- [5] Kieran Alden, Paul S Andrews, Fiona AC Polack, Henrique Veiga-Fernandes, Mark C Coles, and Jon Timmis. Using argument notation to engineer biological simulations with increased confidence. *Journal of the Royal Society Interface*, 12(104):20141059, 2015.
- [6] Mohamed El-Attar, Hamza Luqman, Peter Karpati, Guttorm Sindre, and Andreas L Opdahl. Extending the uml statecharts notation to model security aspects. *IEEE Transactions on Software Engineering*, 41(7):661–690, 2015.
- [7] Hajo A Reijers, Thomas Freytag, Jan Mendling, and Andreas Eckleder. Syntax highlighting in business process models. *Decision Support Systems*, 51(3):339–349, 2011.
- [8] Gregor Jošt, Jernej Huber, Marjan Heričko, and Gregor Polančič. Improving cognitive effectiveness of business process diagrams with opacity-driven graphical highlights. *Decision Support Systems*, 103:58–69, 2017.
- [9] Object Management Group (OMG). *Structured assurance case metamodel (SACM)*, Version 2.0, 2018. <https://www.omg.org/spec/SACM/About-SACM/>.
- [10] Richard Hawkins, Ibrahim Habli, Dimitris Kolovos, Richard Paige, and Tim Kelly. Weaving an assurance case from design: a model-based approach. In *High Assurance Systems Engineering (HASE), 2015 IEEE 16th International Symposium on*, pages 110–117. IEEE, 2015.

- [11] Richard Hawkins, Tim Kelly, John Knight, and Patrick Graydon. A new approach to creating clear safety arguments. In *Advances in systems safety*, pages 3–23. Springer, 2011.
- [12] Chiara Picardi, Richard Hawkins, Colin Paterson, and Ibrahim Habli. A pattern for arguing the assurance of machine learning in medical diagnosis systems. In *International Conference on Computer Safety, Reliability, and Security*, pages 165–179. Springer, 2019.
- [13] Robin Bloomfield and Peter Bishop. Safety and assurance cases: Past, present and possible future—an adelard perspective. In *Making Systems Safer*, pages 51–67. Springer, 2010.
- [14] Ran Wei, Tim P Kelly, Xiaotian Dai, Shuai Zhao, and Richard Hawkins. Model based system assurance using the structured assurance case metamodel. *Journal of Systems and Software*, 154: 211–233, 2019.
- [15] Dirk van der Linden, Anna Zamansky, and Irit Hadar. A framework for improving the verifiability of visual notation design grounded in the physics of notations. In *Requirements Engineering Conference (RE), 2017 IEEE 25th International*, pages 41–50. IEEE, 2017.
- [16] Marco Brambilla, Jordi Cabot, Javier Luis Cánovas Izquierdo, and Andrea Mauri. Better call the crowd: using crowdsourcing to shape the notation of domain-specific languages. In *Proceedings of the 10th ACM SIGPLAN International Conference on Software Language Engineering*, pages 129–138, 2017.
- [17] Daniel L Moody, Patrick Heymans, and Raimundas Matulevičius. Visual syntax does matter: improving the cognitive effectiveness of the i* visual notation. *Requirements Engineering*, 15 (2):141–175, 2010.
- [18] Fabiano Dalpiaz, Xavier Franch, and Jennifer Horkoff. istar 2.0 language guide. *arXiv preprint arXiv:1605.07767*, 2016.
- [19] Dirk van der Linden, Irit Hadar, and Anna Zamansky. What practitioners really want: requirements for visual notations in conceptual modeling. *Software & Systems Modeling*, pages 1–19, 2018.
- [20] Thomas RG Green, Ann E Blandford, Luke Church, Chris R Roast, and Steven Clarke. Cognitive dimensions: Achievements, new directions, and open questions. *Journal of Visual Languages & Computing*, 17(4):328–365, 2006.
- [21] Gennaro Costagliola, Andrea Delucia, Sergio Orefice, and Giuseppe Polese. A classification framework to support the design of visual languages. *Journal of Visual Languages & Computing*, 13(6):573–600, 2002.

- [22] Maria das Graças da Silva Teixeira, Glaice Kelly Quirino, Frederik Gailly, Ricardo de Almeida Falbo, Giancarlo Guizzardi, and Monalessa Perini Barcellos. Pon-s: a systematic approach for applying the physics of notation (pon). In *Enterprise, Business-Process and Information Systems Modeling*, pages 432–447. Springer, 2016.
- [23] Harald Störrle and Andrew Fish. Towards an operationalization of the "physics of notations" for the analysis of visual languages. In *International Conference on Model Driven Engineering Languages and Systems*, pages 104–120. Springer, 2013.
- [24] Jeannette Stark, Richard Braun, and Werner Esswein. Systemizing colour for conceptual modeling. 2017.
- [25] GSN. *Goal Structuring Notation (GSN) Community Standard*, Version 2, 2018. <https://scsc.uk/r141B:1?t=1>.
- [26] John Rushby, Xidong Xu, Murali Rangarajan, and Thomas L Weaver. Understanding and evaluating assurance cases. 2015.
- [27] Timothy Patrick Kelly. Arguing safety: a systematic approach to managing safety cases, 1998.
- [28] Y. Matsuno and S. Yamamoto. An implementation of gsn community standard. In *1st International Workshop on Assurance Cases for Software-Intensive Systems (ASSURE)*, pages 24–28, May 2013. doi: 10.1109/ASSURE.2013.6614267.
- [29] Adelard. *Claims, Arguments and Evidence (CAE)*. <https://www.adelard.com/asce/choosing-asce/cae.html>.
- [30] Robin Bloomfield and Peter Bishop. *Safety and Assurance Cases: Past, Present and Possible Future – an Adelard Perspective*, pages 51–67. Springer London, London, 2010. ISBN 978-1-84996-086-1. doi: 10.1007/978-1-84996-086-1_4. URL http://dx.doi.org/10.1007/978-1-84996-086-1_4.
- [31] Iain Bate, Richard Hawkins, and John McDermid. A contract-based approach to designing safe systems. In *Proceedings of the 8th Australian workshop on Safety critical systems and software-Volume 33*, pages 25–36. Citeseer, 2003.
- [32] Richard Hawkins, Kester Clegg, Rob Alexander, and Tim Kelly. Using a software safety argument pattern catalogue: Two case studies. In *International Conference on Computer Safety, Reliability, and Security*, pages 185–198. Springer, 2011.
- [33] Tim Kelly and John McDermid. Safety case patterns-reusing successful arguments. 1998.
- [34] P. Bishop, R. Bloomfield, L. Emmet, C. Jones, and P. Froome. *Adelard Safety Case Development Manual*. Adelard, 1998.

- [35] R. Bloomfield and K. Netkachova. Building blocks for assurance cases. In *2014 IEEE International Symposium on Software Reliability Engineering Workshops*, pages 186–191, Nov 2014. doi: 10.1109/ISSREW.2014.72.
- [36] Peter Bishop, Robin Bloomfield, J Penny, and A. Eaton. A methodology for safety case development. In *Safety-Critical Systems Symposium*, 1998.
- [37] Takuya Saruwatari and Shuichiro Yamamoto. D* framework creation procedure from collaboration diagram. *IT CoNvergence PRACTice (INPRA)*, 2(2):43–54, 2014.
- [38] Takuya Saruwatari, Shuichiro Yamamoto, and Yutaka Matsuno. A comparative study of d*framework and gsn. *2013 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*, pages 315–320, 2013.
- [39] Toshinori Takai and Hiroyuki Kido. A supplemental notation of gsn aiming for dealing with changes of assurance cases. In *Software Reliability Engineering Workshops (ISSREW), 2014 IEEE International Symposium on*, pages 461–466. IEEE, 2014.
- [40] Michitaro Okano and Takako Nakatani. *A Study on the OR Decomposition of Goal-Oriented Analysis Using GSN*, pages 643–657. Springer International Publishing, Cham, 2014. ISBN 978-3-319-11854-3. doi: 10.1007/978-3-319-11854-3_56. URL http://dx.doi.org/10.1007/978-3-319-11854-3_56.
- [41] M. Flood and I. Habli. Multi-view safety cases. In *6th IET International Conference on System Safety 2011*, pages 1–6, Sept 2011. doi: 10.1049/cp.2011.0260.
- [42] Patrick John Graydon. Towards a clearer understanding of context and its role in assurance argument confidence. In *International Conference on Computer Safety, Reliability, and Security*, pages 139–154. Springer, 2014.
- [43] John Krogstie, Guttorm Sindre, and Håvard Jørgensen. Process models representing knowledge for action: a revised quality framework. *European Journal of Information Systems*, 15(1):91–102, 2006.
- [44] Peter Wegner and Dina Goldin. Interaction as a framework for modeling. In *Conceptual Modeling*, pages 243–257. Springer, 1999.
- [45] Luc Engelen and Mark van den Brand. Integrating textual and graphical modelling languages. *Electronic Notes in Theoretical Computer Science*, 253(7):105–120, 2010.
- [46] X. He, Z. Ma, W. Shao, and G. Li. A metamodel for the notation of graphical modeling languages. In *31st Annual International Computer Software and Applications Conference (COMPSAC 2007)*, volume 1, pages 219–224, 2007. doi: 10.1109/COMPSAC.2007.27.

- [47] David S Wile. Abstract syntax from concrete syntax. In *Proceedings of the 19th international conference on Software engineering*, pages 472–480, 1997.
- [48] Uwe Breitenbücher, Tobias Binz, Oliver Kopp, Frank Leymann, and David Schumm. VINO4TOSCA: A visual notation for application topologies based on Tosca. In *OTM Confederated International Conferences "On the Move to Meaningful Internet Systems"*, pages 416–424. Springer, 2012.
- [49] Katrin Arning and Martina Ziefle. "it's a bunch of shapes connected by lines": Evaluating the graphical notation system of business process modeling languages. In *Full paper at the 9th International Conference on Work With Computer Systems, WWCS*, 2009.
- [50] George Popescu and Alain Wegmann. Using the physics of notations theory to evaluate the visual notation of seam. In *Business Informatics (CBI), 2014 IEEE 16th Conference on*, volume 2, pages 166–173. IEEE, 2014.
- [51] Amine El Kouhen, Abdelouahed Gherbi, Cédric Dumoulin, and Ferhat Khendek. On the semantic transparency of visual notations: experiments with uml. In *International SDL Forum*, pages 122–137. Springer, 2015.
- [52] Dirk Van Der Linden and Irit Hadar. A systematic literature review of applications of the physics of notation. *IEEE Transactions on Software Engineering*, 2018.
- [53] Faisal Saleh and Mohamed El-Attar. A scientific evaluation of the misuse case diagrams visual syntax. *Information and Software Technology*, 66:73–96, 2015.
- [54] Mazin Saeed, Faisal Saleh, Sadiq Al-Insaf, and Mohamed El-Attar. Empirical validating the cognitive effectiveness of a new feature diagrams visual syntax. *Information and Software Technology*, 71:1–26, 2016.
- [55] David Granada, Juan Manuel Vara, Marco Brambilla, Verónica Bollati, and Esperanza Marcos. Analysing the cognitive effectiveness of the webml visual notation. *Software & Systems Modeling*, pages 1–33, 2013.
- [56] John Krogstie, Odd Ivar Lindland, and Guttorm Sindre. Towards a deeper understanding of quality in requirements engineering. In *International Conference on Advanced Information Systems Engineering*, pages 82–95. Springer, 1995.
- [57] Daniel Moody and Jos van Hillegersberg. Evaluating the visual syntax of uml: An analysis of the cognitive effectiveness of the uml family of diagrams. In *International Conference on Software Language Engineering*, pages 16–34. Springer, 2008.
- [58] Nicolas Genon, Daniel Amyot, and Patrick Heymans. Analysing the cognitive effectiveness of the ucm visual notation. In *International Workshop on System Analysis and Modeling*, pages 221–240. Springer, 2010.

- [59] Nicolas Genon, Patrick Heymans, and Daniel Amyot. Analysing the cognitive effectiveness of the bpmn 2.0 visual notation. In *International Conference on Software Language Engineering*, pages 377–396. Springer, 2010.
- [60] Patrick Mäder and Jane Cleland-Huang. A visual traceability modeling language. In *International Conference on Model Driven Engineering Languages and Systems*, pages 226–240. Springer, 2010.
- [61] Abdulaziz Algablan. *A Visual Notation and an Improvement for the Syntax of Larman’s Operation Contracts*. PhD thesis, University of Ottawa, 2016.
- [62] Dirk van der Linden, Anna Zamansky, and Irit Hadar. How cognitively effective is a visual notation? on the inherent difficulty of operationalizing the physics of notations. In *Enterprise, Business-Process and Information Systems Modeling*, pages 448–462. Springer, 2016.
- [63] Thomas RG Green. Cognitive dimensions of notations. *People and computers V*, pages 443–460, 1989.
- [64] Ivar Jacobson. *Object-oriented software engineering: a use case driven approach*. Pearson Education India, 1993.
- [65] Carol Britton and Jill Doake. *Object-oriented system development: a gentle introduction*. McGraw-Hill, 2000.
- [66] Grady Booch. Object-oriented development. *IEEE transactions on Software Engineering*, (2): 211–221, 1986.
- [67] Grady Booch. *Object oriented analysis & design with application*. Pearson Education India, 2006.
- [68] Roland Ducournau and Jean Privat. Metamodeling semantics of multiple inheritance. *Sci. Comput. Program.*, 76(7):555–586, 2011.
- [69] Ghan Bir Singh. Single versus multiple inheritance in object oriented programming. *ACM SIGPLAN OOPS Messenger*, 6(1):30–39, 1995.
- [70] Ronald Allen Cain, Janet Andrea De Lu, and Ralph E Lemke. Development system with methods for visual inheritance and improved object reusability, July 22 1997. US Patent 5,651,108.
- [71] Charles P Jazdzewski. Development system with methods providing visual form inheritance, December 14 1999. US Patent 6,002,867.
- [72] Yvonne Rogers, Helen Sharp, and Jenny Preece. *Interaction design: beyond human-computer interaction*. John Wiley and Sons, 2011.

- [73] Alan Cooper, Robert Reimann, David Cronin, and Christopher Noessel. *About face: the essentials of interaction design*. John Wiley & Sons, 2014.
- [74] ISO DIS. 9241-210: 2010. ergonomics of human system interaction-part 210: Human-centred design for interactive systems. *International Standardization Organization (ISO)*. Switzerland, 2009.
- [75] Martin Maguire. Methods to support human-centred design. *International journal of human-computer studies*, 55(4):587–634, 2001.
- [76] M Sheelagh T Carpendale. Considering visual variables as a basis for information visualisation. 2003.
- [77] Irving Biederman. Recognition-by-components: a theory of human image understanding. *Psychological review*, 94(2):115, 1987.
- [78] Robert E Roth. Visual variables. *International Encyclopedia of Geography: People, the Earth, Environment and Technology: People, the Earth, Environment and Technology*, pages 1–11, 2016.
- [79] Albert Atkin. Peirce’s theory of signs. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, summer 2013 edition, 2013.
- [80] Z ITU-T. 151 user requirements notation (urn)–language definition. *ITU-T*, Nov, 2008.
- [81] Marian Petre, Ed de Quincey, et al. A gentle overview of software visualisation. *PPIG News Letter*, pages 1–10, 2006.
- [82] Tatiana De-Wyse, Emmanuel Renaux, and José Mennesson. Using sketch recognition for capturing developer’s mental models. 2018.
- [83] Jeannette Stark, Richard Braun, and Werner Esswein. Perceptually discriminating chunks in business process models. In *2016 IEEE 18th Conference on Business Informatics (CBI)*, volume 1, pages 84–93. IEEE, 2016.
- [84] Paul Harmon and Celia Wolf. Business process modeling survey. *Business process trends*, 36(1):1–36, 2011.
- [85] Nungki Selviandro, Tim Kelly, and Richard David Hawkins. The visual inheritance structure to support the design of visual notations. In *Third International Workshop on Human Factors in Modeling (HuFaMo’18)*. Copenhagen, 2018.
- [86] Chen Ding and Prabhaker Mateti. A framework for the automated drawing of data structure diagrams. *IEEE Transactions on Software Engineering*, 16(5):543–557, 1990.

- [87] ISO 5807:1985. Information processing — Documentation symbols and conventions for data, program and system flowcharts, program network charts and system resources charts. Standard, International Organization for Standardization, Geneva, CH, February 1985.
- [88] OMG. Business Process Model and Notation (BPMN), Version 2.0.2, December 2013. URL <http://www.omg.org/spec/BPMN/2.0.2>.
- [89] Nungki Selviandro, Richard Hawkins, and Ibrahim Habli. A visual notation for the representation of assurance cases using sacm. In *International Symposium on Model-Based Safety and Assessment*, pages 3–18. Springer, 2020.
- [90] Richard Hawkins and Tim Kelly. A software safety argument pattern catalogue. *The University of York, York*, 30, 2013.
- [91] Maurice Flood and I Habli. Multi-view safety cases. 2011.
- [92] Claes Wohlin, Per Runeson, Martin Höst, Magnus C Ohlsson, Björn Regnell, and Anders Wesslén. *Experimentation in software engineering*. Springer Science & Business Media, 2012.
- [93] Thomas Lumley, Paula Diehr, Scott Emerson, and Lu Chen. The importance of the normality assumption in large public health data sets. *Annual review of public health*, 23(1):151–169, 2002.
- [94] Johnny Saldaña. *The coding manual for qualitative researchers*. Sage, 2015.
-