

Towards Process Models for Goal-Based Development of Enterprise Information Systems Architectures

MALIHE TABATABAIE

PhD

Submitted in partial fulfilment of the requirements for the degree of
Doctor of Philosophy

The University of York
Department of Computer Science
United Kingdom

December 2011

Abstract

Enterprises are organisations with multiple business processes; they often use Enterprise Information Systems (EIS) to support these business processes. The concept of an EIS has arisen from the need to deal with the increasingly volatile requirements of modern large-scale organisations. EIS are growing in use and are now being used to support government, health care, and non-profit / non-governmental organisations. The development of EIS has been affected significantly by the complexity and size of enterprises and their business processes, in addition to the influences of economical, social, and governmental factors.

There are many challenges associated with building EIS. Three critical ones identified in the literature are: adequately satisfying organisational requirements; building valid and stakeholder-acceptable business processes; and providing repeatable and rigorous approaches to establish shared understanding of EIS goals. These challenges are difficult to cope with because of the need to deal with different goals, changes in goals, and the problem of how to transform these goals into system requirements and, ultimately, to an EIS architecture.

This thesis contributes a rigorous approach for identifying and describing the enterprise-level requirements of IT developers, managers, and other stakeholders of an enterprise. The approach provides two modelling and tool-supported processes to help establish a rigorous model of EIS goals. It also provides support for transforming goals to a strategic EIS architecture.

The approach presented in the thesis is based on the concepts of Goal-oriented software engineering. The thesis presents a novel Process Model, KAOS- β that extends goal-oriented software engineering approaches with new concepts and techniques for EIS. Further, to support the transition from requirements to an EIS architecture, an EIS Architecture Process Model (EAPM), is designed and evaluated. Using KAOS- β and EAPM in concert provides a rigorous, repeatable and tool-supported approach for analysing, and designing a strategic EIS architecture. The thesis illustrates the approach with two substantial examples from the health informatics and critical systems domain.

Contents

1	Introduction	15
1.1	Motivation	16
1.2	Research Questions	18
1.3	Research Methodology	19
1.4	Thesis Outline	20
1.5	Thesis Contributions	23
2	Literature Review	27
2.1	Enterprise Information System	28
2.1.1	EIS Characteristics	31
2.1.2	EIS Examples	31
2.1.3	EIS Development Challenges	32
2.2	Goal Definitions	34
2.2.1	Goal vs. Requirement vs. Claim	34
2.2.2	Motivation For GOA	36
2.3	Solutions For GOA	37
2.3.1	Suitability Of KAOS For EIS	40
2.4	Process Modelling	44
2.5	EIS Architecture	45
2.5.1	Background	46
2.5.2	Enterprise Architecture Solutions	47
2.5.3	Software Architecture	53
2.5.4	EIS Architecture Review Summary	56
2.6	Definition Of Method	56
2.7	Overview Of Case Study Domains	56
2.7.1	Stroke Care	58
2.7.2	Airport Crisis Management	65
2.8	Hypothesis	68
2.9	Conclusion	69

3	Method: Process Model	71
3.1	KAOS- β	72
3.1.1	KAOS- β Elements	72
3.1.2	KAOS- β Structure	74
3.2	EAPM	77
3.2.1	EAPM Elements	77
3.2.2	EAPM Structure	80
3.3	Conclusion	82
4	Method: Philosophy	83
4.1	KAOS- β Philosophy	83
4.2	EAPM Philosophy	89
4.3	Conclusion	91
5	Method: Tool	93
5.1	Eclipse Process Framework	93
5.2	Conclusion	96
6	Method: Illustration	97
6.1	KAOS- β Example	97
6.2	EAPM Example	108
6.2.1	EAPM: Tasks One and Two	108
6.2.2	EAPM: Task Three	109
6.2.3	EAPM: Task Four	111
6.2.4	EAPM: Tasks Five, Six, and Seven	112
6.3	Conclusion	118
7	The Airport Crisis Management Example	119
7.1	Airport Crisis Management: Background	119
7.2	Applying KAOS- β	120
7.2.1	First Iteration	120
7.2.2	Second Iteration	131
7.3	Applying EAPM	134
7.4	Conclusion	141
8	Evaluation	143
8.1	Evaluation of KAOS- β	143
8.2	Evaluation of EAPM	149
8.2.1	Process Model Standards	149
8.2.2	Process Models Comparison	153
8.3	Conclusion	161

9 Thesis Conclusion	163
9.1 Research Questions Revisited	165
9.1.1 Question One	165
9.1.2 Question Two	166
9.1.3 Question Three	167
9.2 Lessons Learnt	167
9.3 Future Work	168
9.4 Conclusion	169
A Results of Piloting ACM	171
A.1 Step 4: Document goals	171
A.2 Step 6: Document links	175
A.3 EAPM Tables	177
Glossary	183
Bibliography	185

List of Figures

1.1	Thesis methodology, an exploratory approach	20
1.2	Thesis conceptual Model	22
1.3	EIS development roadmap	25
2.1	Domain analysis of EIS	30
2.2	Models in KAOS technology	39
2.3	KAOS Activity Diagram	40
2.4	An example of EIS goal bloat	43
2.5	Software Architecture Activities	55
2.6	Software Architecture Process	55
2.7	The structure of the parent goal and its three child goals . . .	64
3.1	Method to address the gap in enterprise level systems analysis: the context of KAOS- β and EAPM	71
3.2	KAOS- β Elements	73
3.3	KAOS-Beta Process	75
3.4	Presentation of EAPM elements	79
3.5	EIS architecture process	80
4.1	Stroke care EIS modules	86
5.1	Kestrel, a tool to support KAOS- β	94
5.2	EAPM-tool Snapshot	95
6.1	The structure of the parent goal and its three child goals . . .	101
6.2	The structure and refinement of application goal	107
6.3	Business processes, quality attributes, sample strategies	117
6.4	Business process layer of the architecture	117
6.5	Selected strategies in quality attributes and service layer . . .	117
6.6	Partial architecture for stroke care system	117
7.1	The modules identified for ACM	122

7.2	ACM modules for the KAOS- β pilot	123
7.3	Goal Refinement for ACM goals in the first iteration	127
7.4	Links between goals and agents during the first iteration	129
7.5	Second iteration of goal refinement for SCM goals	132
7.6	Links between the goals and agents in the second iteration	133
7.7	Conflicts between goals in the second iteration	133
7.8	Business process diagram for Airport Crisis Management	139
7.9	EIS architecture for ACM using three-tier style	140
8.1	SPEM Activity Diagram	145
8.2	SPEM WorkFlow Diagram	145
8.3	Kestrel, a tool to support KAOS- β	145
8.4	SPEM activity diagram for EAPM	152
8.5	SPEM use case for EAPM	152
8.6	EAPM-tool Screenshot	152

List of Tables

2.1	EIS boundaries, objectives, and challenges	29
2.2	Negative and positive characteristics of SOA	52
3.1	Template for the structured documentation of KAOS- β 's goals	75
3.2	Example of the structured documentation of KAOS- β 's link refinement	76
4.1	Examples of goal features model annotations	85
4.2	Comparing KAOS and the new KAOS- β process	88
6.1	Structured documentation for goal with ID: SCGT1	99
6.2	Structured documentation for goal with ID: SCGT11	99
6.3	Structured documentation for goal with ID: SCGT12	99
6.4	Structured documentation for goal with ID: SCGT13	100
6.5	Structured documentation for goal with ID: SCGL20	102
6.6	Structured documentation for goal with ID: SCGL201	103
6.7	Structured documentation for goal with ID: SCGL21	103
6.8	Structured documentation for goal with ID: SCGL211	104
6.9	Structured documentation for goal with ID: SCGL212	104
6.10	Structured documentation for goal with ID: SCGL213	105
6.11	Structured documentation for goal with ID: SCGL22	105
6.12	Structured documentation for goal with ID: SCGL221	106
6.13	Structured documentation for goal with ID: SCGL222	106
6.14	Relationship between goals and quality attributes	108
6.15	Template for EAPM results	114
6.16	A filled example of EAPM results template (Table 6.15) . . .	115
7.1	Structured documentation for ACM's goal with ID:ACMG1 .	124
7.2	Structured documentation for ACM's goal with ID:ACMG10 .	125
7.3	Structured documentation for ACM's goal with ID:ACMG2 .	125
7.4	Structured documentation for ACM's goal with ID:ACMG3 .	126
7.5	Documentation for the link between ACMG1 and ACMG2 . .	128

7.6	Documentation for the link between ACMG1 and ACMG3 . .	128
7.7	Goals that indicate the existence of agents	129
7.8	Architectural information for reliable communication quality attribute	134
7.9	Architectural information for safe communication quality attribute	136
7.10	Architectural information for reliable DB quality attribute . .	137
7.11	Architectural information for usability quality attribute	138
8.1	Evaluation of KAOS- β against Ramsin's criteria	148
8.2	Results of comparing EAPM and PALM.	155
A.1	Structured documentation for ACM's goal with ID:ACMG4 . .	172
A.2	Structured documentation for ACM's goal with ID:ACMG5 . .	172
A.3	Structured documentation for ACM's goal with ID:ACMG6 . .	173
A.4	Structured documentation for ACM's goal with ID:ACMG7 . .	173
A.5	Structured documentation for ACM's goal with ID:ACMG8 . .	174
A.6	Structured documentation for ACM's goal with ID:ACMG9 . .	174
A.7	Documentation for the link between ACMG1 and ACMG4 . .	175
A.8	Documentation for the link between ACMG1 and ACMG5 . .	175
A.9	Documentation for the link between ACMG2 and ACMG6 . .	175
A.10	Documentation for the link between ACMG2 and ACMG7 . .	175
A.11	Documentation for the link between ACMG5 and ACMG8 . .	176
A.12	Documentation for the link between ACMG5 and ACMG9 . .	176
A.13	Architectural information for safe database quality attribute . .	177
A.14	Architectural information for different communication devices quality attribute	178
A.15	Architectural information for supporting different interface quality attribute	178
A.16	Architectural information for availability quality attribute . .	179
A.17	Architectural information for modifiability quality attribute . .	179
A.18	Architectural information for performance quality attribute . .	180
A.19	Architectural information for cost-benefit and time to market quality attribute	181

Acknowledgements

I thank my supervisors, Prof. Richard Paige and Dr. Fiona Polack, for their invaluable guidance, support and encouragement throughout my doctoral research.

I would like to thank my colleagues and countless friends in the department and outside for their support and friendship. I thank my colleagues and friends in the Enterprise System group for many interesting and entertaining discussions.

I would like to express gratitude to my parents, my sisters Maryam and Homa, and my brother, without whom this journey was not possible. I would like to dedicate this thesis to those who never stopped supporting me, Maryam, Susan, Matthew, Martin, Andrea, and many more friends.

Author's Declaration

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

Parts of the work presented in this thesis have been published as a referenced book chapter, and as referenced conference and workshop papers:

- Malihe Tabatabaie, Richard F. Paige, Christopher Kimble. *Exploring the boundaries of Enterprise Information Systems*, in Proc. 2nd York Doctoral Symposium, YDS 2008, York, UK, October 2008
- Malihe Tabatabaie, F.A.C. Polack, Richard F. Paige. *Evaluating Goal-Oriented Analysis in the Domain of Enterprise Information Systems*, in Proc. 1st International Conference on Enterprise Information Systems, CENTERIS 2010, Springer, Porto, Portugal, September 2010
- Malihe Tabatabaie, F.A.C. Polack, Richard F. Paige. *KAOS- β : Analysing EIS architecture using KAOS*, in Proc. 8th International Workshop on Modelling, Simulation, Verification and Validation of Enterprise Information Systems, ICEIS 2010, Madeira, Portugal, June 2010
- Malihe Tabatabaie, Richard F. Paige, Christopher Kimble. *Exploring Enterprise Information Systems*, in Enterprise Information Systems for Business Integration in SMEs: Technological, Organizational and Social Dimensions, IGI Global, 2009

Chapter 1

Introduction

An Enterprise Information System (EIS) is a software platform capable of supporting and integrating activities across an organisation (Wikipedia, 2011d). The range of EIS applications is growing; they are now being used to support government, health care, non-profit / non-governmental organisations, and commercial enterprises (Stair & Reynolds, 2006; Bowers, 2010; Harrell, 2009). The UK National Health Service information technology portfolio (NHS) is a working example of an EIS (Morse, 2011). Other examples include the EIS that integrates different business processes in Mitsubishi (Mitsubishi Logistics, 2011); and an Airport Crisis Management (ACM) system (International Airport Review, 2011; MODELPLEX Consortium, 2007) that integrates different business processes for dealing with emergency situations.

The concept of an EIS has evolved to improve organisational coordination, efficiency, decision making, and address organisational requirements (Laudon & Laudon, 2007). These enterprise-level requirements include functional and non-functional requirements for software systems, reducing costs of development processes, dealing with distributed systems, sharing business processes, and making best use of resources (Tabatabaie, Paige, & Kimble, 2010).

The increasing demands for the use of EIS, and the challenges involved in their deployment and development, are the original motivation for research in this domain. The main challenges of developing an EIS are categorised as follows.

Complexity: A complex system is non-deterministic, where its behaviour cannot be predicted by analysing component interactions (Wegmann, 2003). An EIS is a system that deals with organisations and human interactions. Even though organisations are structured entities, the ex-

pectations of a system that supports the functionalities and Business Processes of an organisation cannot be predicted just by analysing component interactions. The interactions between human users and business processes within an organisation are complex, and the results cannot be predicted by analysing individual components. The development process of an EIS should identify and address the elements of complexity.

Business processes: EIS are used to support the inter-connected business processes of an organisation. Each business process is designed to address dynamic organisational demands and values (Tabatabaie et al., 2010). There is a challenge in developing an EIS that satisfies the functionalities of business processes and supports changes in business processes as well (Stair & Reynolds, 2006). This challenge extends to further difficulties in capturing the knowledge of business processes and required functionalities.

The aim of this thesis is to develop a rigorous approach to capture the volatile and strategic level requirements of EIS and to help make a transition from this knowledge to the EIS design phase. The approach is supported by systematic processes and tools.

1.1 Background and Motivation

Since the 1950s, organisations have been developing computer-based information systems (IS) to support their business processes. Through improvements to IS and changes in the ways that businesses use IS, computer-based systems have become more complex and yet more reliable; at the same time, increasing functional requirements have been placed on these systems (Edward et al., 1993).

Even though the technology used in computer-based IS has become more reliable, the literature reveals many cases of computer-based IS that fail to support their businesses, functionalities or business changes. Some examples are summarised below.

Consider the case of BMW. “The BMW Group is reputed to be the only manufacturer of automobiles and motorcycles worldwide that concentrates single mindedly on premium manufacturing standards on outstanding quality for all its brands” (Stair & Reynolds, 2006). This company failed to handle a change in the market demand for their Z3 and X5 products (Stair & Reynolds, 2006). As a result, the BMW production line received some of the parts while other parts were still under development. They could not produce the cars

till all the parts arrived, hence large storage rooms were required for arriving parts; this caused extra expense. This was explained by the company's use of an IS that does not provide functionalities to support a tight and coordinated supplier network (Stair & Reynolds, 2006). An EIS with such functionalities would have facilitated the capture of knowledge, such that decision makers may have become aware of changing demand patterns in BMW's partners. Thus, they would not have increased the production line (change) in a way that did not match with the partners' capacities. This failure is an example of the challenges caused by EIS complexity: new demand and changing business processes.

The Hilton Hotels example appears much earlier in the literature¹. In 1992, a reservation system for hotels and cars was requested by four major partners, including the Hilton Hotels Corporation. The development team failed to deliver the requested system. Two of the cited reasons for this failure are project mismanagement and changing goals (Keef, 2011). One of the key challenges of developing an EIS is dealing with changes in the market and resulting business processes. In this case, the changes occurred in the early phases of development, particularly goal identification. Explicit and structured goals for an EIS are a source of knowledge that could be shared between decision makers and IT developers. As an EIS is a complex system, its development could suffer from unclear or unrealistic goals and expectations. Addressing all the goals of an EIS might be impossible but an agreed set of preliminary goals is the first step towards developing an EIS. This example illustrates the complexity of developing EIS that could be caused by the changes in the goals, demands and values of an enterprise.

The UK National Health Service (NHS) electronic patients' records and online booking system is another example of complex and challenging EIS (DH Stroke Policy, 2007). The size, the number of business processes, and large number of stakeholders makes the development process very challenging. Some of the functionalities of this EIS are sensitive and could lead to risk to human life. This EIS was expected to be implemented in 10 years from its inception in 2002 (with an allocated budget of £12.7 bn). Because of these challenges, it is predicted that the expected deadline will be extended. In the 8th year (2010), only thirteen out of 169 acute trusts² received the full system but half of the budget was spent (Bowers, 2010). Even the implemented parts were suffering from incidents; for example, in 2006, during

¹<http://www.computerworld.com/computerworld/records/images/pdf/44NfailChart.pdf>

²"Hospitals are managed by acute trusts. Acute trusts make sure that hospitals provide high-quality healthcare and that they spend their money efficiently" (Acute Trusts, 2011). Acute trusts employ the medical and non-medical staffs for NHS.

four months of system operation, more than 110 major incidents (e.g. failure of x-ray retrieval hardware and software, patients records and planned treatments) hit hospitals across England (Collins, 2006). In 2008, thousands of electronic patients' records were lost for no clear reasons (Bowers, 2010). The human interaction, the size and impact of the NHS in the society, in addition to the organisational complexity of NHS are challenges that EIS need to address.

In all these examples, organisations, regardless of their size and line of work, hired IT experts to develop or purchase a suitable IS that addresses their business processes. The main question is: *why even well-planned and experienced IT groups fail to deliver the required systems as promised?*

The Bull survey in 1998 (Bull, 1998) reveals twelve reasons for IT projects failure in the finance sector. Two major reasons are *bad communication between relevant partners* and *lack of planning and scheduling resources and activities*. These reasons are discussed further in Chapter 2 and leads to defining focused motivation for the rest of this thesis. The motivation for this thesis is the lack of existing process models for identifying and capturing knowledge of EIS demands and values. These processes could help partners communicate and plan the use of resources and activities and develop a shared understanding of business processes.

To address this motivation a number of research questions have been identified.

1.2 Research Questions

This thesis is motivated by a key question: *why does the process of building an EIS fail to deliver its functionalities and address users' goals?* The focus of the thesis is on understanding the difficulties within the development process, rather than on implementation difficulties such as an absence of required devices or errors in code.

By considering different types of EIS examples and analyses of EIS, which are available in the literature review (see Chapter 2), the thesis develops specific questions that focus on concrete EIS challenges. This process is called domain analysis. The concept was first introduced in (Neighbors, 1980) as a method for identifying elements such as the objects and operations of a software class.

The EIS domain analysis leads to the following research questions:

1. What are the essential characteristics of an EIS?

2. Why does an EIS fail to deliver its functionalities and fail to address stakeholders' and enterprises' goals?
3. What knowledge is required to identify the essential functionalities for an EIS?

Chapter 2 starts with an analysis of the literature. It identifies a number of gaps that are derived from the research questions. The rest of this thesis is organised so as to answer the three major research questions, using the research methodology introduced in the next section.

Nomenclature: in this thesis, keywords are in italic format and capitalised. They are summarised in the glossary section.

1.3 Research Methodology

To address the research questions in Section 1.2, a number of qualitative and quantitative approaches in the domain of Software Engineering (SWE), *Requirements Engineering (RE)*, and Human Computer Interaction (HCI) have been reviewed and analysed. The results of the review illustrate that there are few, if any, techniques available to support the transformation of enterprise-level goals to a strategic EIS architecture.

This thesis develops a systematic approach to the transformation of enterprise goals and requirements, into a strategic EIS architecture. To do this, an exploratory methodology is used, to support the transitive characteristics of the results and approaches used in this thesis. The results of each phase initiate the techniques and approaches used for the next phase. The iterative software engineering (SWE) process models are based on the idea of accepting the changes, reviewing the results in each phase and advancing to the next phase; and these characteristics are the motivations for using iterative SWE process models for this thesis.

One of the process frameworks that could be extended and adapted to create more specific software engineering processes is *evolutionary development* (Sommerville, 2007). "Evolutionary development is based on the idea of developing an initial implementation, exposing this to user comment and refining it through many versions until an adequate system has been developed" (Sommerville, 2007). To address the development of software systems with unclear problem domain specifications, the theory of evolutionary development may be applied. Using this theory the development team can iteratively improve their implementations until they are satisfied with the results.

Figure 1.1 summarises the steps of the exploratory approach used in the thesis. An exploratory methodology starts with a literature survey. Even though research questions motivate the study, a domain analysis clarifies the problems, boundaries, and the elements of the domain. A literature survey phase leads to understanding the problem and collecting data to start the research (step 1). Indeed, the data collection phase and analysing the data would help to re-evaluate the problem and if required, re-define it. The data collection (step 2), leads to designing a model to address and possibly solve the problem (step 3). A model also could provide a new perspective to address the problem and the domain of the problem. To analyse and evaluate a model it should be applied to examples from the domain (step 4). The results of the empirical analysis helps to evaluate the model and the problem (step 5). As can be seen in Figure 1.1 this methodology is based on the concept of iteration. Iterations continue until there is confidence in the collected data and results. However, time and resources also limit the explicit and implicit iterations.

In Section 1.4, the relationships between the exploratory methodology and the research model that is followed for this thesis is presented.

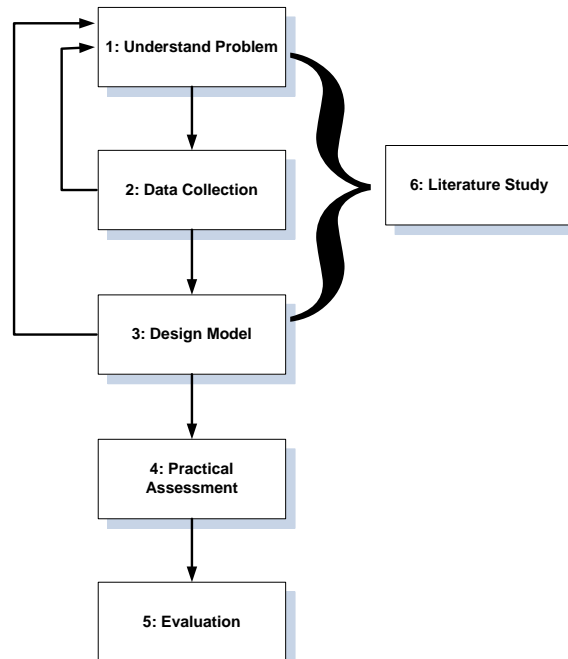


Figure 1.1: Thesis methodology, an exploratory approach (Sommerville, 2007).

1.4 Thesis Outline

This section presents the blueprint of the thesis and its accomplishments, aligned with the conceptual model presented in Figure 1.2.

The thesis starts with domain analysis (box 1 in Figure 1.2); this phase is aligned with the step 1 in Figure 1.1. The thesis then identifies the gaps in the literature associated with developing EIS (box 2 in Figure 1.2); this phase is aligned with step 2 in Figure 1.1. The gap that this research is focusing on is *understanding, specifying, and describing diverse and volatile stakeholders' goals*.

To address the identified gap, a general review on the current methodologies and technologies related to developing EIS has been undertaken, leading to detailed review and analysis of Goal-Oriented Approaches (GOA) (box 3 in Figure 1.2). The analysis of GOA illustrates two main gaps in this domain in respect of enterprise goal analysis: *lack of clear process guidance for GOA* and *strong dependency on technologies' experts and domain experts*. This analysis leads to the development of a process model, KAOS- β , to address the identified gaps (box 4 in Figure 1.2); this phase is aligned with step 3 in Figure 1.1. The relationship between the KAOS- β and EIS goals in Figure 1.2 illustrates that KAOS- β could be used to identify EIS goals (box 5 in Figure 1.2).

To demonstrate the effects of identifying EIS goals on the later stages of the EIS design process, this research focuses on tracing the results of the KAOS- β to an EIS architecture. Strategic EIS architecture is a challenging area because of the lack of a clear definition. To address the challenges and propose solutions, this area has been the subject of much research. As a result, commercial solutions such as DODAF and MODAF have been developed to help develop EIS architectures. This thesis reviews and analyses these solutions to identify the process models used in the current solutions (boxes 7 and 8 in Figure 1.2). The analysis of EIS architecture and related technologies leads to developing a novel process model, EAPM (box 6 in Figure 1.2); this phase is also aligned with step 3 in Figure 1.1. The transition from KAOS- β to EAPM is illustrated in Figure 1.2 and leads to identifying a number of quality attributes for EIS (box 9 in Figure 1.2). These quality attributes could be used as an early information for designing an EIS architecture (box 10 in Figure 1.2).

To evaluate the results of the process models developed in this thesis, Chapter 7 presents the phases of developed process models to a new example of EIS, ACM. This phase is aligned with steps 4 and 5 in Figure 1.1.

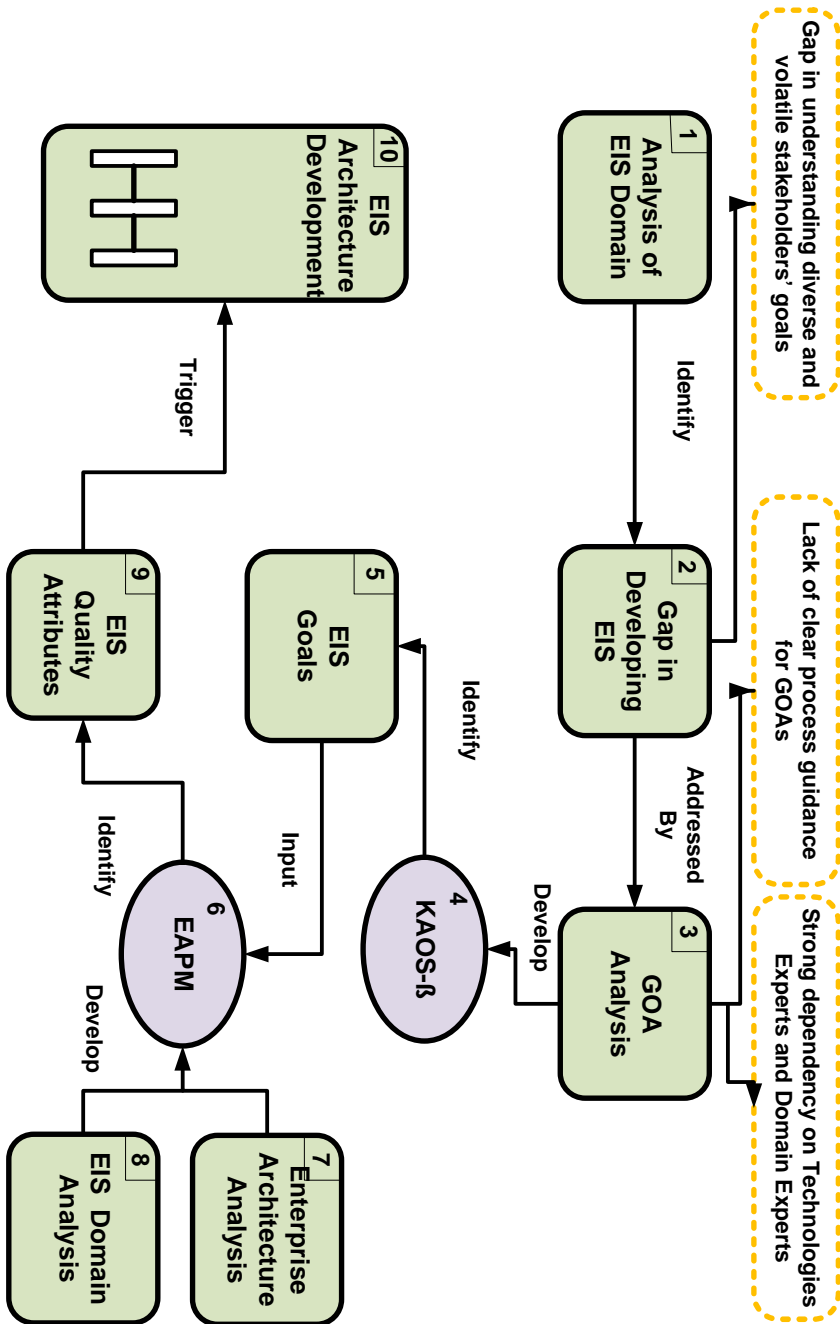


Figure 1.2: Thesis conceptual Model

1.5 Thesis Contributions

This section summarises the contributions of this thesis, and their impact on addressing the research questions.

Central Contribution: The key contribution of this thesis is a novel method that develops goals for Enterprise Information Systems and traces them through to a strategic EIS architecture. This method includes two novel processes, KAOS- β and EAPM.

KAOS- β : The first process model, KAOS- β , provides steps to analyse, identify, and structure the goals for an EIS. This new process model adapts and extends KAOS (a requirement engineering method) to the enterprise context. The results of KAOS- β could be used to support decision making, during design phases (e.g. architectural design), or eliciting requirements of an EIS.

EAPM: The second process model, EAPM, is developed here to address the strategic level of developing an EIS architecture. EAPM provides a novel perspective for developing an EIS architecture using software architecture concepts and technologies, and building on the output of KAOS- β .

Process Model Development: Two process models (KAOS- β and EAPM) are developed in this thesis. The steps and approach towards developing these process models are explicitly explained and analysed. This systematic approach could be used as guidance for developing other process models in the domain of software engineering.

Process Model Evaluation: There are no widely accepted approaches for determining the quality of a specified process or process model (Alegría, 2011). This thesis introduces techniques that could be used for evaluating a process model and a novel demonstration of applying and use of these techniques for evaluating KAOS- β and EAPM. These evaluation techniques could be used for similar process models in the domain of software engineering.

EIS Definition: There is no standard definition of EIS. This thesis presents a domain analysis of EIS, including a summary of EIS characteristics, that provides an explicit definition for EIS (Section 2.1).

In summary, the contributions of this thesis address the three research questions presented in Section 1.2 as follows:

1. *What is an EIS and its characteristics?*

To address this question, the EIS and its domain has been analysed an EIS definition, its characteristics, examples of what is and what is not an EIS, are discussed in this thesis.

2. *Why does an EIS fail to deliver its functionalities and address stakeholders' and enterprises' goals?*

The domain analysis illustrates that unclear goals in the early phases of decision making and design, causes different understandings of how an EIS should integrate and support the business processes. Therefore, goal-oriented approaches in the domain of software engineering are analysed and a process model to address one possible approach is developed and tested.

3. *What knowledge is required to identify the required functionalities for developing an EIS?*

Analysing EIS examples in the literature illustrates that each development phase requires specific knowledge. This knowledge should be shared and understandable for different parties. To identify and develop required functionalities, we propose to trace the original goal knowledge to the further development phase. In this thesis, the goal knowledge is traced to the architecture design knowledge.

Figure 1.3 presents the use of this thesis' contributions, KAOS- β and EAPM, in the development process of an EIS. The process starts by the initial analysis of the EIS domain and uses the KAOS- β to identify EIS goals. The identified goals are used to identify requirements in an iterative process. The identified goals also are used to design an EIS architecture by providing initial information for EAPM. EAPM leads to identifying a number of EIS quality attributes that trigger the EIS architecture.

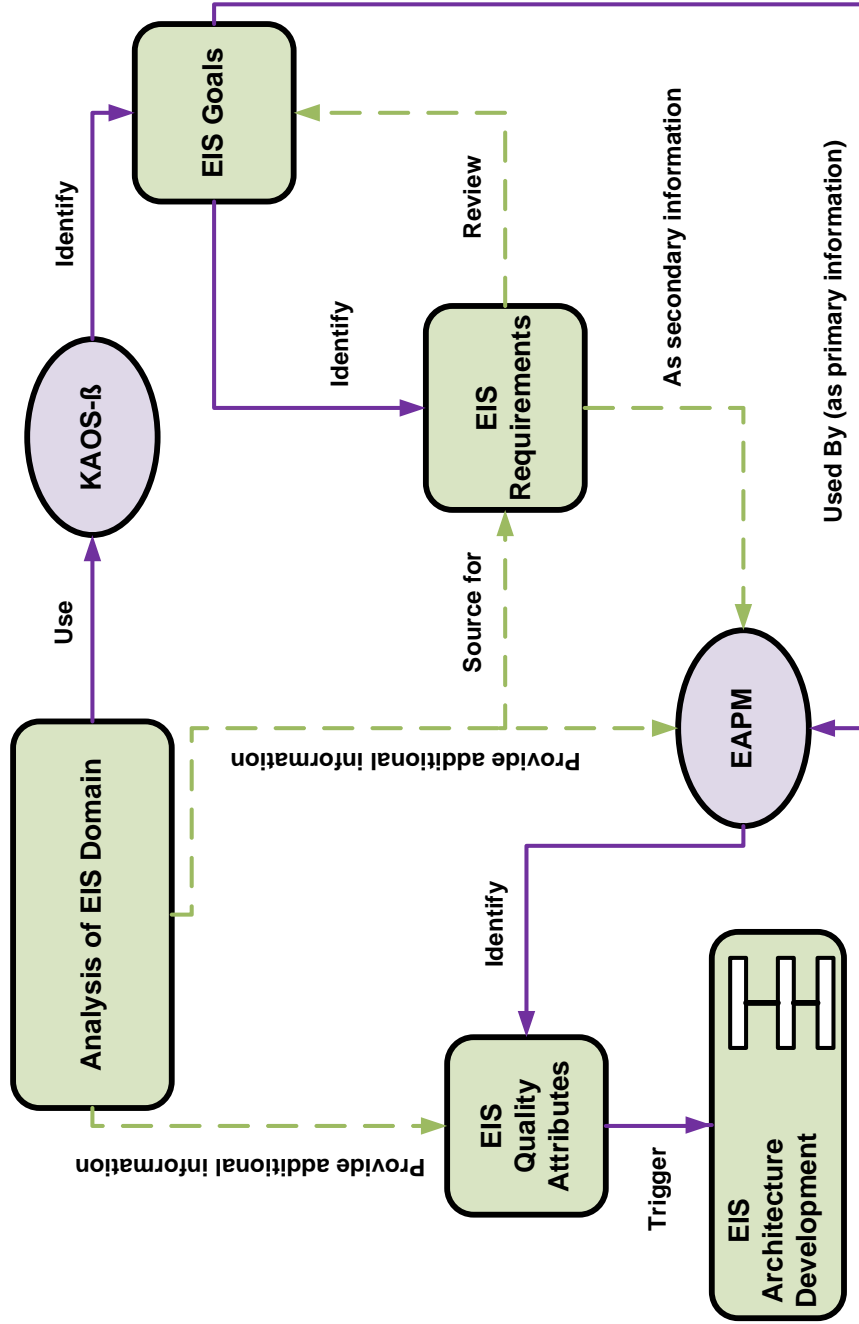


Figure 1.3: EIS development roadmap. Solid lines present primary links and dashed lines present secondary links.

Chapter 2

Literature Review

The aim of Chapter 2 is to present a review and analysis of the literature relevant to the development of EIS. We use the review to clarify and precisely define the characteristics and challenges of developing an EIS. The review considers research from software engineering, and their applicability to EIS development challenges.

Section 2.1 presents a discussion review of the EIS domain, including EIS definition, characteristics, examples and challenges. Sections 2.2 and 2.3 present a review of Goal-Oriented Approaches (GOA), including a definition of goal and related concepts, in addition to an empirical review of a selection of successful GOA. This review identifies a number of limitations for GOA; one limitation is the lack of clarity in the underlying process models for GOA. This limitation leads to a review of process modelling approach in Section 2.4.

Some of the EIS development challenges presented in Section 2.1 motivate an investigation of EIS architecture. Therefore, Section 2.5 presents a review of a number of architectural methods in the domain of EIS.

Because the contributions of this thesis are presented in the form of a method for developing EIS, Section 2.6 presents a definition and brief review of the constituent elements of a method. To help answer the research questions of this thesis, we carry out two EIS case studies; these case studies are introduced in Section 2.7. EIS case studies introduction illustrates the rationale to adopt these cases. Finally, the results of EIS literature review leads to identifying a number of gaps.

Section 2.8 presents focused gaps and a testable hypothesis to address these gaps. Finally, Section 2.9 summarises the results of Chapter 2 and introduces the structure of this thesis's chapters to address the hypothesis.

2.1 Enterprise Information System

The notion of an enterprise system was established after the First World War, when new industries came to the market and many industries combined and amalgamated (Fruin, 1994). (Fruin, 1994) identified three types of enterprises that share common elements such as inter-firm relations, marketing, mode of competition, finance, ownership, management, administrative coordination, government relations.

Mitsubishi is an example of an enterprise dating back to 1926; it integrates distinct yet affiliated companies, particularly Mitsubishi Heavy Industry, Mitsubishi Warehousing, Mitsubishi Trading, Mitsubishi Mining, Mitsubishi Bank, Mitsubishi Electric, Mitsubishi Trust, Mitsubishi Property, Mitsubishi Steel, Mitsubishi Oil, Nippon Industrial Chemicals, and Mitsubishi Insurance (Fruin, 1994). Another enterprise example is General Electric, which has independent divisions focusing on health care, aviation, oil and gas, energy, electrical distribution, security, and many others (Electric, 2011).

EIS is a large-scale information system that supports the business processes of an enterprise. The examples of Mitsubishi and General Motors illustrate that an enterprise is composed of several business processes and multiple groups of stakeholders. Therefore an EIS must support multiple business processes and related stakeholders. Technologies such as *Service Oriented Architecture (SOA)* are currently popular in design and implementation of systems to support multiple businesses processes, platform, and information systems. The term business process often implies a process that focuses on delivering financial value. However, in practice, large-scale business processes, and their associated information systems support delivery of different kinds of outcome, which are not always directly linked to financial values. In fact, businesses include both financial organisations and public organisations, which deliver services to the public.

By analysing published definitions, organisation, and business model characteristics (Tabatabaie et al., 2010), we adopt the following definition of EIS:

An Enterprise Information System is a software system that integrates the business processes of organisation(s) to improve their functioning.

Integration of business processes plays an important role in this definition. Integration can be accomplished by providing standards for data and business processes. These standards are applied to various part of the system such as a database or clusters of databases. One aim of integration is to allow data required by more than one business process to flow seamlessly.

Another point in this definition is the software characteristics of EIS. At this stage, we consider EIS as a type of information system; therefore an EIS may include humans, software, and hardware systems.

The next term used in the definition is organisation. Organisations may include an organisation with its partners, or a group of organisations. Table 2.1 refines the above definition and describes what we propose as the objectives, goals, domain, and challenges of EIS (Tabatabaie et al., 2010; Laudon & Laudon, 2007).

EIS Objective	Integrating business processes of an organisation
	Seamless information flow
	Suitable access to data and information for various stakeholders
	Matching the software system structure with organisation structure and requirements
EIS Goal	Improving coordination, efficiency, and decision-making of business process in an organisation
EIS Domain	Covers the internal and external business activities of organisation
Particular EIS Challenges	Security challenges that should be considered carefully for organisations' processes.
	Managing integrated information needs of multiple business processes and changing enterprise goals.
	Mixing the required information of one business process with another one can cause problem for the organisation
	Improve flexibility in organisation processes

Table 2.1: EIS boundaries, objectives, and challenges.

Figure 2.1 gives a graphical presentation of the EIS definition. An enterprise may contain one or more organisations (represented as large triangles). Organisations have their own business processes (BPx), which have their own data requirements. The EIS provides standard information technology (IT) support across all the business processes, interfacing with the databases, warehouses, and other computer systems of the enterprise. This structure allows co-ordination of IT systems across business processes, allowing data sharing and supporting security policies, as well as providing flexibility needed to support enterprise level strategies. Beyond the technology, an EIS needs to support EIS strategy, i.e. providing suitable information for management to make decisions.

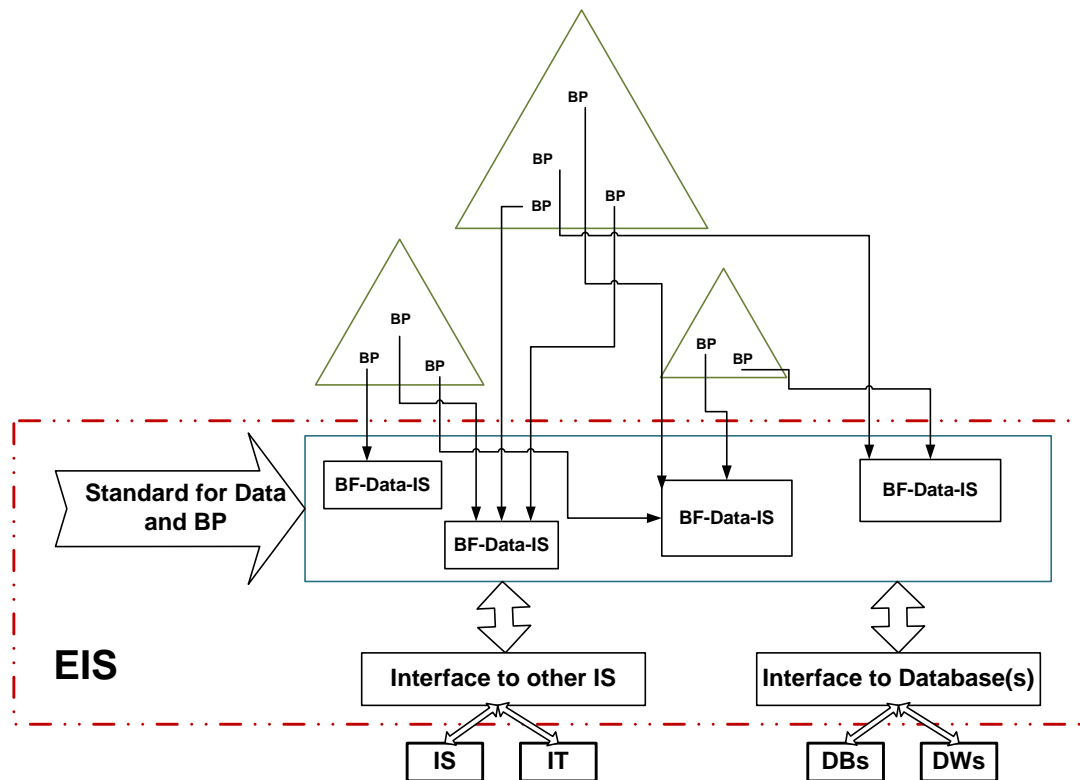


Figure 2.1: Domain analysis of EIS model. BP in this figure are business processes and BF are business functions; based on (Tabatabaie et al., 2010).

2.1.1 EIS Characteristics

To identify the defining characteristics of an EIS, a number of technologies and terminologies have been reviewed (Tabatabaie, 2011): system of systems, adaptive systems, legacy systems, distributed systems, service-oriented architecture, catalysis approach, and the IBM component business model (Tabatabaie, 2011).

The analysis identifies the following characteristics for an EIS:

- Supports and orchestrates multiple business processes
- Supports organisational goals
- Supports changes and evolution of an enterprise functionalities
- Dynamic architecture to accommodate and integrate new systems
- Contains sensitive and real-time data and processes
- Open system: includes interactions with systems (e.g. human, hardware, software systems)

None of these characteristics is based on the size of the organisation; an EIS covers different sizes of enterprises, small, medium, or large. It is also not essential for an EIS to be geographically distributed, though the fundamental nature of enterprise makes this common.

2.1.2 EIS Examples

The definition of EIS that was presented previously can help to distinguish an EIS from other information systems. To refine the distinction, some examples are now provided.

Mitsubishi has more than 400 companies around the world (Mitsubishi Committee, 2011) and multiple business process is an enterprise (See section 2.1). An information system that links various parts of the Mitsubishi organisation (including high-level managers) and supports seamless collaborations between business processes would be an EIS.

The infrastructure needed to support the National Health Service (NHS), can be considered as an EIS. Here, information systems are being developed to support management of patient records and prescriptions. Such infrastructure aims to connect currently independent departments within and outside of the NHS, whilst implementing a strict security policy. E-Government is another public sector enterprise that could use an EIS, to connect various

governmental organisations or departments together to let information flow seamlessly between them.

Counter-examples can further clarify the definition of enterprise and EIS. Online shops such as eBay (Gopalkrishnan & Gupta, 2007) and Amazon (Wikipedia, 2011a) do not have the requisite diversity of business processes: they are not enterprises, and their organisational needs can be supported by conventional information systems.

There are also organisational systems development projects that do not in themselves qualify as an EIS. A classic example is the 1980s attempt to develop an automated ambulance dispatch system for London Ambulances Service (LAS) (Finkelstein, 1993). Whilst ambulance dispatch is a process of the wider NHS EIS, this system was not design to integrate with patient and health care management. It addresses a single strategic goal, rapid response to emergency call outs- it did not, for instance, incorporate forward planing of casualty treatment. Whilst many of the issues that arise in this project relate to failure to appreciate the complexity of the problem, the LAS is not an EIS by the criteria identified in this thesis.

This thesis focuses on two enterprises and their EIS (more information is given in Section 2.7). The stroke care example takes a strategic level view of stroke management and prevision in a hypothetical health service (based on documented examples of UK (DH Stroke Policy, 2007), US (Schwamm, Pancioli, Acker, Goldstein, Zorowitz, Shephard, Moyer, Gorman, Johnston, Duncan, Gorelick, Frank, Stranne, Smith, Federspiel, Horton, Magnis, & Adams, 2005), and Denmark (Harrell, 2009)). This qualifies as an enterprise, because it covers many business processes with a need for integrated data management, e.g. patient admission, emergency treatment and its records, rehabilitation services, etc.

The Airport Crisis Management (ACM) example (MODELPLEX Consortium, 2007) deals with strategic information and decisions in crisis situations. It requires an EIS because it must integrate, capture, and disseminate information from crisis site to air side and land side of the airport, among staff and passengers, press, emergency services etc.

2.1.3 EIS Development Challenges

An EIS supports business processes of an enterprise. Therefore the challenges of developing EIS are associated with the challenges of developing complex information system, multiple enterprise business processes, and enterprise structure (i.e. different stakeholders and point of view).

(Berg, 2001) reviews the concept and reasons for the success and failure of developing health care information systems. The results of (Berg, 2001)

review applies to EIS as EIS is a complex information system and face some of the challenges of complex health care systems.

This review highlights some of the challenges of complex systems applied to EIS as an EIS is a complex system too.

An early challenge, discussed in (Berg, 2001) is the design issues that leads to failure in implementations. There are different aspects of design for an EIS, including early design and planing, and architecture design. Challenges in each of these phases could lead to failure in the final result. For example, according to (Armour et al., 1999), a mis-perception of EIS architecture design that leads to failure is to start the development from detailed architectural design rather than providing a blueprint for creating enterprise-wide information system.

The challenges in design are also influenced by lack of clear criteria and measure for success and failure of EIS (Berg, 2001). (Berg, 2001) argues that there is no simple formula to define success factors, even when there is a total agreement on the goals. This highlights the importance of defining goals of an EIS to gather further information about an EIS development, which leads to goal identification challenges. (Berg, 2001) also emphasises that conflicts between goals and policy and regulations is another challenge for developing EIS. In addition, a set of goals and policy, while is successful in a system, might not be successful in another one.

Another challenge highlighted by (Berg, 2001) originates from a characteristic of EIS that was discussed in Section 2.1.1: Involving multiple business processes and different stakeholders' point of view. The failure to manage processes that address different points of view is a common challenge of developing an EIS (Berg, 2001).

In addition to these challenges, several other common challenges for complex information systems has been presented by (Bull, 1998; Royal Academy Engineering, 2004) that could be categorised as an EIS challenges. For example, lack of clear communication between relevant partners (Bull, 1998; Royal Academy Engineering, 2004); inadequate resources and skills to deliver the total portfolio (Royal Academy Engineering, 2004); too little attention to breaking development and implementation into manageable steps (Royal Academy Engineering, 2004). These are examples of implementation and technical challenges.

In summary, implementation challenges such as inadequate resources, skills, and technologies have been in centre of attention for developing an EIS. The aim of this thesis is to focus on design challenges to investigate systematic processes for developing an EIS, in addition to providing traceable design solutions in advance of implementing a complex EIS that requires

substantial amount of resources.

One of the challenges that is presented in EIS examples is identifying and structuring goals of an EIS, which leads to extracting further design and implementation information (Berg, 2001; Liu et al., 2000). Section 2.2 presents a review of the goal concept and technologies. The design challenges also motivate this thesis to further the investigations to review EIS architecture and the influence of identifying goals on the EIS architecture in Section 2.5.

2.2 Goal Definitions

In section 2.1.3 some of the EIS challenges of early phases of development have been reviewed. (Liu et al., 2000) argues that organisational and enterprise long-term goals must be taken into consideration in developing an EIS. This suggests that goal analysis is a useful approach in addressing part of the design challenges. Thus, the objective of this section is to precisely define the notion of *goal* that is used for this thesis, and to review existing GOA. GOA have been widely discussed in the requirement engineering domain (van Lamsweerde, 2001, 2004). GOA are also used in the safety and security research community – for example, to present safety cases and safety arguments (Kelly, 2004; Kelly & Weaver, 2004) – and in software assessment (Weiss et al., 2002).

Even though the notion of goal is used in different GOA, there is no general agreement on a standard definition. For example, the term goal used in GSN presents the concept of an arguable claim, whereas in the requirements engineering domain, a goal is a system goal, leading to one or more system requirements. Section 2.2.1 presents a review of goal definition in comparison with other similar concepts. Later in Chapter 3 we narrow down the choices of GOA to the ones that are aligned with this thesis definition of goal. Nevertheless, to learn from the successful aspects of practical GOA, the concepts of GOA in general and in particular a number of successful approaches are reviewed in Section 2.3.

2.2.1 Goal vs. Requirement vs. Claim

The terms goal and requirement are often used interchangeably; the notion of claim is also used simultaneously in argumentation (Kelly, 1998a). This section discusses the differences between these terminologies.

(Antón, 1996, p. 137) defines goals as “high level objectives of the business, organization, or system. Goals express the rationale for proposed systems and guide decisions at various levels within the enterprise”. This defi-

inition is used as a primary goal definition for this thesis. (Kim et al., 2006) define a goal model, similarly to (van Lamsweerde, 2003), as a criterion for designing the architecture for systems, used to guide decisions at various levels; source of requirements, software, and data architecture. (van Lamsweerde, 2003) uses goals for identifying requirements and as source for deriving data and software architecture.

(Sommerville, 2007, p. 118) defines requirements as “the descriptions of the services provided by the system and its operational constraints”. The concrete difference between requirements and goals is that the requirement inquires *what* needs to be implemented whereas goal inquires *why* and *how*. Goals indicate what needs to be achieved, that leads to identifying requirements (Regev & Wegmann, 2005; Nuseibeh & Easterbrook, 2000; Darimont et al., 1997).

We assume that by answering what we expect a system to achieve, and *Why*, a number of functionality and requirements are defined, a set of goals would be collected. (Antón, 1996, p. 137) supports this argument by defining the relationship between goals and requirements as follows: “a requirement specifies how a goal should be accomplished by a proposed system”. A requirements document is developed with the developer team in mind; thus a requirements document may be unclear for the stakeholders, but the goal document should be understandable and agreed on for both stakeholders and developers.

In short, the definition of goal that is used in this thesis is aligned with (Antón, 1996); goals are collected from domain analysis and should be understandable to the users and decision makers. Requirements are technical and, compared to goals, include more detailed information that could be used by system developers to understand the details of the system design and implementation.

Claim is another interpretation of notion of goal. A claim is a statement intended to be demonstrated to be true; hence it could be the motivation for developing an argument, to show that it is true or false. A claim could be defined based on the description of defined goals, objectives of a system, requirement, or developed functionalities of a system. An example of a goal is: the system should satisfy the users by providing suitable response to their queries. An example of a requirement is: the system should respond to the user’s queries in less than 10 seconds. An example of a claim is: the system provides a response to the user in less than 10 seconds.

Knowledge of what constitutes a goal is the first step towards applying a GOA. Based on our definition of goal, the next section gives a rationale for using a GOA as the basis for developing an EIS.

2.2.2 Motivation For Goal–Oriented Approaches

An enterprise is a collection of businesses processes, and sometimes business partners that operate as an organisation with shared goals. An EIS is a software system that integrates support for the business processes of organisation(s) to improve their functioning (Tabatabaie, Paige, & Kimble, 2008; Tabatabaie et al., 2010). The challenges of developing and deploying EIS vary across enterprises, with many challenges related to scale and complexity. Some factors that create complexity in EIS are (Tabatabaie et al., 2010):

- Growth of the size of information systems and enterprises
- Interactions between different information systems
- Involvement of many different parties in the construction and use of information systems
- Ever-increasing rate of organisational and social change

Involvement of different parties in the construction and use of information systems and the social, technical, organisational changes brings the challenges of dealing with diverse and volatile stakeholders' goals. In addition to the lack of precise approach to bridge the gap between stakeholder understanding and IT system expert understanding of an EIS structure. These factors also lead to failure in communication and understanding of organisational requirements, identified as challenges earlier (RAE-BCS Working Group, 2004). Techniques that can help to address these challenges could help to reduce the risks inherent in developing and deploying EIS.

To develop an EIS that satisfies the stakeholders' expectations of a system, the ideal case is to involve them in the process of developing an EIS. At this stage, however, we cannot expect business experts to design their own software systems but we can provide support for them in identifying and structuring their goals; similar support could be provided for EIS developers. Through a structured process of analysis and modelling goals, EIS developers are led towards an understanding of the priorities of the enterprise and its organisational goals. Analysis requires identification of a sufficiently-complete set of goals, by engagement with different groups of stakeholders. Engagement with senior staff can help to create understanding between developers and organisational seniors. Structuring the goals of EIS can lead towards a vision of the EIS-to-be and systems-as-is, and indirectly addresses the challenge of EIS structural visualisation.

2.3 Solutions For Goal–Oriented Approaches

Based on the previous review of goals and motivation for GOA, Section 2.3 summarises current GOA. We do this to identify features of existing GOA that are of value in EIS development.

The need to identify enterprise–level goals led us to review well–known GOA techniques, especially in the domain of requirements elicitation and engineering. Of the GOA reviewed, some have previously been applied to EIS, but all are aimed at use by GOA experts, rather than domain experts. A review of a number of GOA is presented in Section 2.3. The reviewed GOA do not generally include clear process of how to extract and refine the goals and goal structures: this is a critical concern for EIS where overlapping, contradictory, and possibly inconsistent goals may lead to enterprise-wide problems. There are various types of EIS with very different requirements, and it may therefore not be possible to develop a generic process that can satisfy all of them. Approaches tailored for specific EIS domains may be more profitable. However, in any case, when there is no process, the use of an approach is limited by its reliance on GOA experts or personal interpretation by its users.

There are at least 15 distinct GOA (Kavakli & Loucopoulos, 2005), in areas such as artificial intelligence (Hong, 2001), software assessment (Weiss et al., 2002), requirements engineering (van Lamsweerde, 2009; Castro, Kolp, & Mylopoulos, 2002) and safety argumentation (Kelly, 1998b, 2004).

In this section we report our review of four leading approaches by applying them to an EIS example, stroke care (Tabatabaie, Polack, & Paige, 2010a). The approaches, which were selected after considering the definition of EIS, the aim of the research, and the internal characteristics of each GOA, are Goal Structuring Notation (GSN) (Kelly, 1998b), KAOS (van Lamsweerde, 2009), GBRAM (Antón, 1996), and i^* (Castro et al., 2002). KAOS and GBRAM have approaches for defining and refining the goals of an organisation; KAOS is the better-documented approach, and has been used in many successful projects. i^* is a well-known and successful approach, and has been applied to health care (An et al., 2009); it claims to encourage the involvement of stakeholders in requirement analysis, and to help the developers to achieve a deep understanding of the domain. GSN is widely used for presenting the structure of arguments in the domain of safety, and has been applied to requirements analysis in research contexts (Habli et al., 2007).

Critique of i^* : “The i^* framework (i^* stands for distributed intentionality) views processes as involving social actors who depend on one another for goals to be achieved, tasks to be performed, and resources to be

furnished” (Yu et al., 1996). i^* captures implicit yet important information for business processes and IS that support them. i^* supports process modelling and re-engineering (Yu & Mylopoulos, 1994). The motivation of i^* is to understand why a business process is the way it is, rather than just to describe the requirements for a business process. i^* was chosen for review because of its focus on understanding of the business environment and the domain (Yu & Mylopoulos, 1994); its encouragement of stakeholder involvement in requirements analysis; its visual notation for communication between stakeholders; and, finally, its previous use in health care (An et al., 2009).

However, like (An et al., 2009), we find that i^* requires significant detailed information early in the design process. Whilst the level of detail may exist in business-specific IS development, it is not appropriate for analysis of enterprise goals. This problem applies in some part to the other GOAs, but is particularly problematic in i^* , which requires detailed information for allocating dependency relationships, and produces a goal structure that includes implementation information.

Critique of GSN: The critique of GSN is based on (Tabatabaie, 2009).

GSN is a notation for presenting an argument, and one of the requirements to start an argument is a claim about the system. “A claim is a statement that you are asking the other person to accept” (Toulmin, 1958). For example a claim would be that a system is safe. GSN uses a set of notations, including modules, goals, strategies, context, and assumptions to support the claims. Even though using claim is successful for the GSN, it is not match with the goal definition of this thesis. Similarly, because of the origin of GSN origins, goals are derived from safety requirements and expressed as a claim over evidence; by contrast, in most requirements engineering GOA, goals are expressed before identifying the requirements and are a source for identifying the requirements (Kavakli et al., 1996; Antón, 1996; van Lamsweerde, 2009).

GSN is typically used to summarise an argument over evidence of safety, from traditional safety analysis techniques. Whilst GSN provides a powerful structuring notation, it does not provide a goal identification method which is one of the requirements for this thesis.

Because of its origins in safety-case argumentation, the GSN notation includes useful features such as modularity or context, but cannot easily express contradiction or priority of goals, perhaps because they are not needed for an argument. In EIS, it is invariably the case that the

enterprise assigns priorities to its goals, and it is generally the case that some identified goals will conflict (excellence vs cost-saving is a common instance).

Critique of GBRAM: (Antón, 1996) presents GBRAM as an approach for analysing, identifying, and classifying goals, *Agents* and stakeholders. The detail level of GBRAM is more appropriate for identifying goals than that of i^* : goals are a “logical mechanism for identifying, organizing and justifying software requirements” (Antón, 1996). GBRAM provides a top-down approach to refining and structuring goals; it addresses identification and documentation of goals, as well as the issue of knowing when the goals are adequately specified. These positive attribute influence our eventual approach.

The main disadvantage of GBRAM is its lack of a generic analysis process: this lack of guidance inhibits use the method as-is for EIS.

Critique of KAOS: KAOS is a GOA, designed to elicit and validate requirements and to prove their consistency (Delor et al., 2009). Objectiver is a tool that supports KAOS. The developers of the Objectiver tool¹ state that KAOS extends the traditional “what question” approach to requirements with “why”, “who” and “when” questions.

KAOS develops a *goal model*, from which other models can be derived: the *obstacle model*, the *agent model*, the *operation model*, the *object model* and the *behaviour model*. Figure 2.2 presents the relationships between these models that cover different views and aspects of the system requirements.

Even though KAOS does not have an explicit process model for structuring a goal model, it provide a number of heuristics. KAOS has many conceptual similarities to GBRAM (Objectiver, 2010). It has both a top-down approach and a bottom-up approach to identifying and refining goals. In this sense, it is more detailed and complete than GBRAM; KAOS’ approach is also at an appropriate level for EIS, without the need for detailed information early in the analysis that is problematic in i^* . The rigorous approach to goal definition in KAOS can cause difficulties for stakeholders who are unfamiliar with IT development, but adds its own value in respect of concrete definition.

The KAOS methodology is actively evolving and is well-documented; it is applied in many industrial cases. Although, as in other approaches, the process of applying KAOS is not well-defined, largely comprising usage examples

¹www.objectiver.com

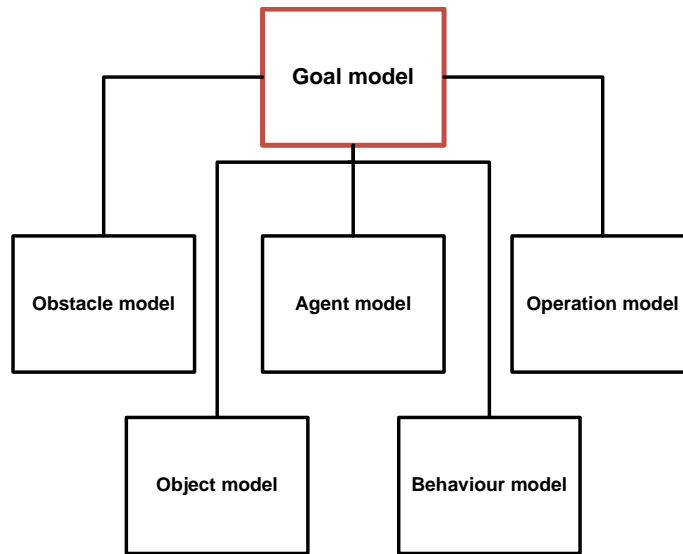


Figure 2.2: Relationship between KAOS models; image is based on the description given in (van Lamsweerde, 2009).

(van Lamsweerde, 2009, p. 502), the advantage of documentation makes it the basis for our EIS process. The overall review of KAOS illustrates the benefit of this methodology compare to the ones we review in this thesis. Therefore KAOS is used as the main methodology for the further study of GOA for this thesis.

2.3.1 Suitability Of KAOS For EIS

Section 2.3.1 considers the aspects of KAOS that are and are not suitable for EIS development.

The analysis of KAOS heuristics focuses on the heuristics used to identify goals. These are summarised in an activity diagram, Figure 2.3.

The first heuristic relates to goal identification. It proposes review of documentations related to the system and its environment for specific goal-related keywords. This heuristic proposes further goal identification through instruction of goal categories.

Some of the goal-related keywords and categories proposed by KAOS (van Lamsweerde, 2009) are not generally applicable to EIS analysis, because the categories aim to finally define requirements for a system and not an EIS and they assume there exist documents that clearly present objectives of a system.

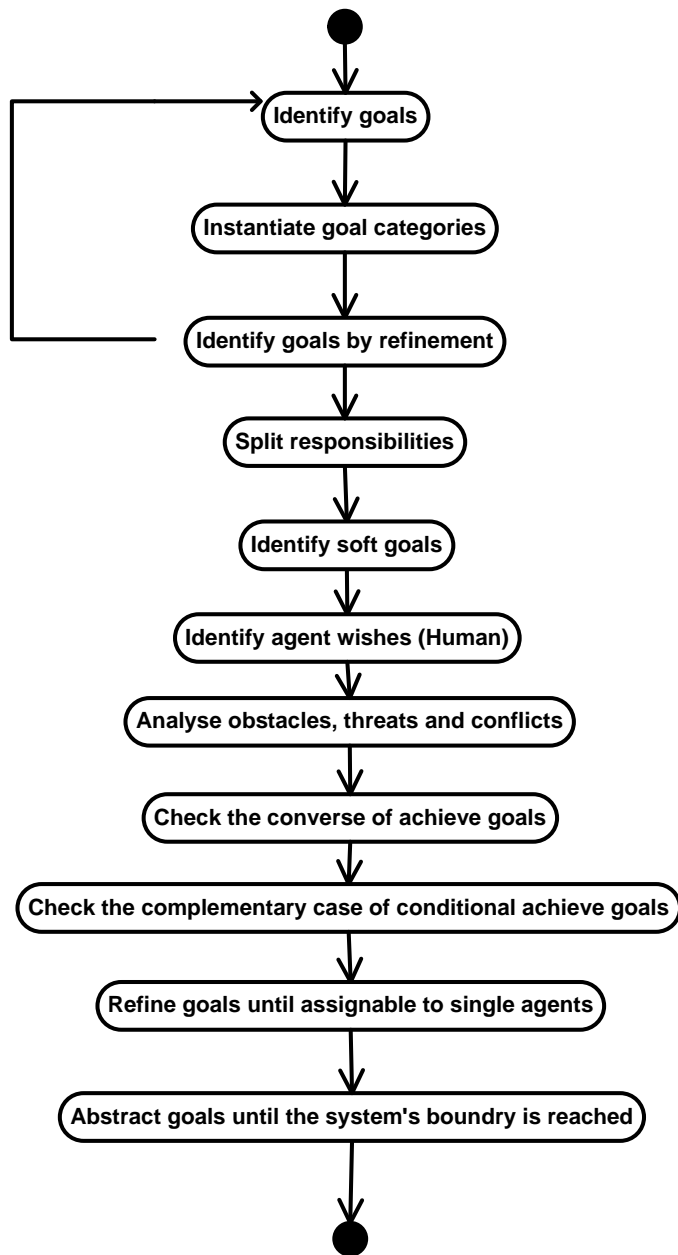


Figure 2.3: Activity Diagram presenting the steps of KAOS methodology (van Lamsweerde, 2009).

Instead, a review of documentation of identified functionalities, plans, and expectations of the EIS leads to identifying part of EIS goals. Further goals were identified from interviews of professional users and researchers.

The KAOS heuristics for identifying goals by using refinement was useful in exploring relationships among goals. KAOS proposes top-down and bottom-up review, posing “How” and “Why” questions, respectively (van Lamsweerde, 2009). The approach is particularly useful in clarifying ambiguous goals and relationships.

The remaining heuristics all pose problems for EIS goal modelling; because EIS goal modelling addresses business processes, and does not relate to a specific development of a single system. Agents also are not identified during enterprise goal analysis.

This means that the KAOS heuristics that proposes to *split responsibilities* (van Lamsweerde, 2009) (i.e. to assign goals to agents) are not applicable to EIS. The same problem appears in a later step: refine goals until applicable to a single agent. The heuristic: *identify soft goals*, is problematic for several reasons: EIS goals modelling does not focus on a single system development, so soft goals that can be used to “prescribe preferences among alternative system behaviours” (van Lamsweerde, 2009, p. 268) cannot be accurately determined. Furthermore, the precise meaning of soft goals is not elaborated in KAOS documentation.

Identify agent wishes (human) is not completely applicable to EIS. However, the goal identification heuristics do reveal human wishes in relation to enterprise strategies and expectations for the EIS.

The heuristics: *analyse obstacles, threats and conflicts*; check the converse of achieved goals; and check the complementary case of conditional achieved goals are all useful guidelines but in the case of EIS goal analysis, the identifies goals are not usually specific enough for these forms of checking. The heuristics *refine goals until assignable to single agents* already been rejected because EIS goal analysis does not identify agents. However, the heuristics are also inappropriate in EIS, since many agents would typically be involved in satisfying any particular EIS goals.

Finally the heuristic: abstract goals until the system boundary is reached, is only applicable in principle. In practice, the boundaries of an EIS are unclear, and as a result do not provide a useful stopping criterion.

The review of KAOS heuristics on stroke care EIS illustrates that some of the heuristics are not directly applicable to EIS. As the result, the Objectiver tool is not fully applicable for our case of EIS. However, by modifying the steps, the Objectiver tool could be used to create KAOS goals model. Objectiver was applied to the stroke care case study (Section 2.7). The resulting

goal model (e.g. Figure 2.4) is complex and poorly structured- it illustrates the phenomenon of goal bloat. Goal bloat is similar to the concept of code bloat (Stevens et al., 2005; Wikipedia, 2011e) that leads to unnecessary complexity in presenting the structure of goals. The complexity of the KAOS goal structure for an EIS highlights the problems associated with applying GOA design for single system requirement analysis to an enterprise. This is exacerbated by KAOS reliance on a single goal model, with no modularity. Benefits of modularity is one of the lessons learnt from analysing GSN and will be used in developing the method of this chapter.

In summary, we identified at least 15 distinct GOA (Kavakli & Loucopoulos, 2005). Existing GOA address system-level goals, so need adaptation for EIS-level goal analysis. The review of the four GOA illustrates a general key limitation which is the lack of result evaluation techniques. Evaluating the results relies heavily on the domain expertise. We also observed that the processes introduced by these techniques on how to apply them are either inapplicable fully to EIS (i.e. GSN, KAOS) or do not exist (i.e. i^*). However, the heuristics of KAOS have the potential to be encoded to a suitable process. To make this transformation, the elements and requirements of a process should be considered. To investigate the elements of a suitable process for a GOA tailored to developing EIS, Section 2.4 presents a review of processes and process models.

2.4 Process Modelling

In 1992, research on software processes started to appear (Feiler & Humphrey, 1993). A process, and in particular, a software process requires human involvement; hence processes are mainly dependent on the final users and the domain experts for their implementation and evaluation. In the domain of software engineering, a process is a “set of partially ordered steps intended to reach a goal” (Feiler & Humphrey, 1993). (Fuggetta, 2000, p. 28) defines a process as a “coherent set of policies, organisational structure, technologies, procedures, and artifacts that are needed to conceive, develop, deploy, and maintain a software product”. Hence processes could be used in different stages of software development to focus on a particular aspect. Consequently process model is an abstract representation of a process architecture, design, or definition (Feiler & Humphrey, 1993). Examples of processes and process models are Business Process Modelling (Mitra, 2008), Application Development Process Support (ADPS), Catalysis, Rational Unified Process (RUP) (Chroust et al., 2010), OpenUp (EPF Project, 2010).

A software process model provides principles and guidelines by providing a precise course of action, tools, procedure, constraints, policies; and it presents a direct correlation between quality of the process and the quality of the developed software (Fuggetta, 2000). A benefit of defining an explicit software process is to evaluate improve the results by enhancing the processes (Fuggetta, 2000; Feiler & Humphrey, 1993). Software processes also identify the different dimensions of software development and the problems need to be addressed (Fuggetta, 2000). Other benefits of an explicit process model are providing the ability to reuse and repeat the process, it is easier to validate as one can analyse elements and structure of a process model. It also makes the process model effective as an explicit process model is easier to document. *Hence providing an explicit process model could improve the quality of a software product.*

Process models are currently presented in natural, formal language, and graphical notations (Chroust et al., 2010). The main characteristics, properties, and requirements of a process model are: activities that have to be accomplished to achieve the process objectives, roles of the people, structure and the nature of the artifacts (Fuggetta, 2000; Feiler & Humphrey, 1993; SPEM, 2009; Chroust et al., 2010). Example of tools, standards and process modelling languages that are currently used are: Eclipse Process Framework (EPF Project, 2009), Software and Systems Process Engineering Metamodel Specification (SPEM) (SPEM, 2009), Business Process Modeling, Business Process Execution Language (van der Aalst, 2003; White, 2006).

Process models have two aspects: methodological and technologi-

cal (Fuggetta, 2000). Technological process models have direct affect on implementation of projects, however this thesis focuses on the methodological aspect of process models by defining the elements, structure, and philosophy of a process model (Chroust et al., 2010). The method is presented in the following chapter.

The importance of process model evaluation has been discussed in the literature, however, the approaches to evaluate process model is an ongoing research. There are attempts to evaluate a process model quantitatively and based on metrics (Garcia et al., 2006; Aguilar et al., 2006). However, these approaches are still far from practical use. External validity of a process model is particularly difficult because generalising the conclusion of the study outside the context where the study was carried out is not possible in most cases (Fuggetta, 2000). To address these issues, a number of criteria (Ramsin, 2006; Chroust et al., 2010; Fuggetta, 2000; Feiler & Humphrey, 1993) and a standard (SPEM, 2009) have been identified to be the basis for evaluating this thesis process models.

So far this section has presented a review of processes, process models, and their elements and standards. This review is the basis for part of the solutions, suggesting GOA, to address the design challenges presented in Section 2.1.3. Another aspect of the design challenges highlights the EIS architecture issues that could lead to failure. Therefore, Section 2.5 presents a review of the enterprise and EIS architecture to investigate the potential solutions for designing an EIS architecture.

2.5 EIS Architecture

So far, goals have been identified as a key element for strategic-level EIS development. The objective of this section is to present another element of EIS strategic-level development: EIS architecture. The influence of goals on data and software architecture has been the subject of research for at least the last decade (Clements & Bass, 2010a; van Lamsweerde, 2003). However, none of them has presented an analysis of the influence of goals on EIS architecture. New solutions are suggested to design enterprise architecture, that are not exactly the same as EIS architecture. This section presents a review of the the current processes and approaches in designing EIS architecture from two perspectives: enterprise architecture (EA) and software architecture.

2.5.1 Background

(McGovern et al., 2003, p. 6) state: “To consider what enterprise architecture means, it is important to understand its origin. All architecture within information technology can trace its ancestry back to the lessons learned from building architecture”. The term architecture has an established definition in the building construction domain. However, in the domain of IT, the term, architecture is not defined so clearly.

Early work on aspects of architecture in the domain of EIS describe different terminologies and technologies such as Enterprise Architectures (EA), Enterprise Information System Architecture (EISArch), System Architecture (SysA), Software Architecture (SWA), Hardware Architecture (HWA), Network Architecture. The difference in the architecture terminology is dependent on the users’ point of view and the domain the architecture is applied to. For example Network Architecture is about the nodes of a network and the relationship that these nodes have on each other (Perry & Wolf, 1992, p. 2); HWA presents the “configuration of architectural pieces of the hardware” (Perry & Wolf, 1992, p. 2). Likewise, the SWA is about the structure that includes the software elements, its visible properties, and the relationship between the elements (Bass, Clements, & Kazman, 2003, p. 3). All these architecture definitions define common concepts such as the kind of components and the connections between them. However, unlike the other architecture terminologies discussed so far, the SysA is “a process and a discipline to produce efficient and effective information systems” (McGovern et al., 2003, p. 1). The main difference between SysA and SWA/HWA/Network architecture is the use of processes and process models in designing SysA. As discussed in the first part of this thesis, EIS is a *system* with enterprise characteristics – it is *organisational* and *business* oriented. Thus EIS architecture inherits the characteristic of SysA and uses processes and process models for its design. Little work has been done on process modelling in this area. Two examples are the IT and business alignment technique by (Molinaro et al., 2010) and the matrix based value model² by (Páscoa et al., 2010).

To investigate the current processes and process models for designing EIS architecture, a domain analysis has been undertaken. The results illustrate that there are two main approaches for design EIS. The first approach appeals to EA concepts and related technologies, and the second approach appeals to software architecture solutions (McGovern et al., 2003). Section 2.5.2 presents the results of analysing EA approaches and Section 2.5.3 presents the results of analysing software architecture solutions.

²Value matrix is used to present the relationship between business objectives and business processes.

2.5.2 Enterprise Architecture Solutions

This section presents a review of current solutions for Enterprise Architecture (EA). The term EA tends to refer to ultra large scale and complicated architectures for systems or organisations. The topic has received little attention, therefore limited references are introduced in this section. In this thesis, an alternative definition of enterprise architecture is used: “a coherent whole of principles, methods, and models that are used in the design and realisation of an enterprise’s organisational structure, business processes, information systems, and infrastructure” (Lankhorst, 2005, p. 3).

The main purpose of EA is to create a clear map of the business processes (BPs) and governance principles to support BPs, so that the IT developers understand what a business wants, and can implement systems that satisfy the enterprise goals (Minoli, 2008; Lankhorst, 2005). “The goal of enterprise architecture is to create a unified IT environment (standardized hardware and software systems) across the firm or all of the firms business units” (Minoli, 2008, p. 9). The essential information of the business that could be captured partly in goals is generally considered to be more stable than specific solutions that are found for the problems currently at hand (Lankhorst, 2005).

A model of EA includes business architecture, information architecture, solution architecture, and technology architecture (Minoli, 2008, p. 16). In this model, the business architecture extends business process modelling tools such as the Business Process Modelling Language (BPML). Business architecture is “an architectural formulation of the business function” (Minoli, 2008) and the information architecture is an “architectural formulation of the information function via a data model” (Minoli, 2008). The information architecture presents data modelling tools such as entity relationship modelling. The solution architecture presents code development tools such as CORBA, SOA (Minoli, 2008). The technology architecture states the network design tools, performance modelling tools (Minoli, 2008, p. 18). Each of these architectures is represented by a set of viewpoints. Each viewpoint is associated with a language that can be used to describe systems from that viewpoint (Minoli, 2008, p. 17). The enterprise architecture goes beyond the IT architecture umbrella (which usually is data and infrastructure architectures) (Minoli, 2008, p. 19). The enterprise architecture explicitly exhibit the business processes and agents’ point of view.

The main characteristic of EA is creating a holistic view of an enterprise. This view is a web of relations between organisational structure, products, operations, and technology (Lankhorst, 2005). (Wegmann, 2003) defines enterprise as “an organization of resources, which performs a process”. Hence, he defines EA as a “discipline that deals with the organization of the enter-

prises resources” (Wegmann, 2003). To design an EA, he introduced four different levels for aspects and viewpoints of an enterprise (e.g. IT level and business level). The role of EA is to integrate business and IT by creating traceability between these levels. This theory is the basis for well-known EA solutions such as Zachman, DODAF, MODAF, TOGAF etc. These methods and solutions address the strategic-level or strategic reasoning architecture. This level of architecture focuses on the decision making and planing rather than providing detailed implementation information (Hoogendoorn, Jonker, Maanen, & Treur, 2009). The following present a domain analysis of a selection of EA solutions.

Zachman’s Framework: In 1987, Zachman introduced a framework for large scale and complex information systems that become the first EA framework (Zachman, 1987). His framework is independent of information systems and aims to integrate the interfaces of the systems’ components. By which he will achieve another level of control for these large scale and complex systems.

“Zachman puts an emphasis on describing what exists on each level of an enterprise. In the simplest version of the framework, Zachman proposes to describe within each level: what things are involved (data); how things are done (function), where things are done (network). The Zachman framework uses an add-hoc graphical notation. No specific CAD tool support is available” (Wegmann, 2003).

Zachman framework suggests an enterprise architecture based on the users viewpoints. This theory motivates further enterprise architecture framework based on viewpoints such as the framework proposed by US Department of Defense³, UK Ministry of Defence⁴, and Open Group⁵.

DODAF: Department of Defense Architecture Framework (DODAF) is a promising product in the field of enterprise architecture, and a possible approach for our example of EIS: the aim here is to find an understanding of both how DODAF works and what its processes are. Note that clear processes, tools, and guidelines are necessary if non-experts are to use the products.

DODAF is an architectural framework that first appeared in the 1990s with the aim of supporting the development of an interoperable and

³<http://cio-nii.defense.gov/sites/dodaf20/>

⁴<http://www.mod.uk/DefenceInternet/AboutDefence/WhatWeDo/InformationManagement/MODAF/>

⁵<http://www.opengroup.org/togaf/>

cost effective military system using comprehensive architectural guidance (DODAF, 2007). A framework in this case provides “guidance and rules for structuring, classifying, and organizing architectures” (DODAF, 2007, p. 1-6). The architecture for DODAF follows a similar definition of architecture to SWArch (Bass et al., 2003) which is a set of components and connections between the components, as well as functional parts and their rules (DODAF, 2007). The aim of using an architecture, from the DODAF point of view, is to enable better decision making throughout the enterprise (DODAF, 2007). Therefore, DODAF concentrates on providing different views to present the stakeholders’ requirements. An early version of DODAF, Version 1.5, used *Products* as well as views to help the users visualise the architecture data. However, in Version 2.0, the concept of Product is removed and the concept of *Described Model* is added. The focus in the new version is more on the *data model* and three main views: operational view, systems and services view, and technical view. In this version the term view is replaced by *view point* (DODAF, 2009).

DODAF establishes two main architecture layers: presentation and data layers. The presentation layer includes the view points and the data layer includes the data model. To satisfy the goal of having *interoperable* military systems, DODAF accommodates Net-Centric and SOA concepts. SOA is a key element for implementing the Net-Centric objectives, that provide the ability to share information when it is needed, where it is needed, and with those who need it (DODAF, 2007).

DODAF is based on *federated architecture*⁶; the first step of DODAF’s process is to “determine the intended use of the architecture”. In this early stage the stakeholders are told what to expect from the architecture, “what the architecture would accomplish and how it may affect the organisations or system development” (DODAF, 2007, p. 2-3).

Although DODAF focuses on federated architecture and uses SOA and Net-Centric ideas, it is essentially an architectural approach/ strategy. To decide if this strategy is required for designing an enterprise architecture, the designers need to have a general understanding of what goals they want to satisfy, and they need to investigate if this strategy is suitable and cost effective. Techniques such as KAOS- β could help

⁶“It provides a framework for enterprise architecture development, maintenance, and use that aligns, locates, and links disparate architectures and architecture information via information exchange standards to deliver a seamless outward appearance to users” (DODAF, 2007, p. 1-6)

the designers to understand the goals better and make a decision on what strategy is most appropriate.

MODAF: MODAF is an EA framework developed by British Ministry of Defence to support defence planning and change management activities (MODAF, 2010). Similar to DODAF, MODAF provides different views and graphical and textual visualisation of the business area for different groups of stakeholders (MODAF, 2010). In addition to the Operational views, Service-Oriented views, System views, and Technical views, MODAF provides Strategic views, Acquisition views, and a Combined system view. Strategic views address business outcome, Acquisition views addresses dependencies and timelines of the project. By adding these views MODAF emphasises the requirement for presenting the business outcomes and their effect on the implementation of the project in an enterprise architecture.

MODAF is based on DODAF and there is no evidence in the public documents that the early stages of the process are different from the DODAF process. Therefore, the argument regarding the requirement for identifying the quality attributes and architectural drivers that we made for DODAF is applicable to MODAF.

TOGAF: The Open Group Architecture Framework (TOGAF) was developed in 1995. It is based on the Technical Architecture Framework for Information Management (TAFIM), developed by the US Department of Defense (TOGAF, 2009). TOGAF is a set of methods and tools for developing and maintaining enterprise architecture.

Architecture in TOGAF refers to the definition in ISO/IEC 42010:2007. The fundamental organization of a system is embodied in its components, their relationships to each other and the environment, and the principles governing its design and evolution (TOGAF, 2009). The main aspects of architecture in TOGAF are the formal description of the system and components, and the relationships, principles and design guidance in the systems. TOGAF aims to reach the level of implementation from the detailed descriptions (TOGAF, 2009). Therefore, it includes business architecture, information systems architecture, and technology architecture. The TOGAF process is about preparing and initiating the required elements for the architecture. Public documentation focuses on defining the main quality attributes and architectural drivers by analysing the organisations' business, IT, and technological aspects.

SOA: The objective of EA methods that have been reviewed so far is to address the business implementation of an enterprise in general and specifically, the relationship between business processes and IT resources (Wegmann, 2003). Aligned with this objective, SOA provides a solution that depends on the concept of services. Services are the functionalities of a system (McGovern et al., 2003, p. 63). A service is “an implementation of a well-defined piece of business functionality, with a published interface that is discoverable and can be used by service consumers when building different applications and business processes” (O’Brien, Bass, & Merson, 2005, p. 1). Services address the implementation of a system functions and structured their interfaces in an standard way to be able to communicate with other services mainly from other systems.

To understand the positive and negative aspects of using SOA, Table 2.2 presents a selection of SOA characteristics.

From a cost point of view, the effectiveness of SOA for designing EIS depends on how an enterprise allocates its budget, the domain of the system, and the expertise of the developers. Therefore the argument on the cost effective characteristics could have both negative and positive aspects. However, from our point of view, SOA provides a strategy for designing the architecture and developing a system rather than an architectural process. When an architect collects early information, they could decide either to use SOA or any other approach for designing their architecture.

The SOA definition given in (O’Brien et al., 2005, p. 1) emphasises that SOA is not just a system that is built as a set of services, but the SOA uses the services to build a particular system or application; the difference is subtle. The SOA definition highlights the importance of justifying the required services and placing them very carefully where they can add benefit, rather than just choosing a number of possible services and designing an ad-hoc system to combine the services. This leads us again to the importance of having a clear set of goals for understanding and choosing most appropriate services. Hence, to follow the SOA logic, SOA uses Business Process Modelling Notation (BPMN) (Wikipedia, 2011c) to define the business processes and identify the functionalities (Erl, 2010). The defined services would collaborate with each other using Business process Executable Language (BPEL) (Wikipedia, 2011b), which is an executable language. The combination of these techniques and languages produces the architecture of SOA. In this case the architecture is the connection between

Positive Characteristics	Negative Characteristics
Abstract and loosely coupled (O'Brien et al., 2005, p. 4)	Refactor and change when the service is published should be avoided (O'Brien et al., 2005, p. 5)
Modular and layered nature (McGovern et al., 2003)	Vulnerable to network issues because of distributed nature (McGovern et al., 2003)
Achieve reliable level of clustering the servers leads to higher availability (Minoli, 2008; McGovern et al., 2003)	Disrupting the service if any of the provider machines not available as SOA are hosted on many machines (McGovern et al., 2003)
More tolerant of network disruption according to asynchronous communication (McGovern et al., 2003)	"The complex nature of some systems built on an SOA makes it very difficult to mathematically derive SLA parameters (Implementation downside)" (McGovern et al., 2003)
Easier remove or replace the services in the architecture (McGovern et al., 2003)	
Increase corporate agility (Minoli, 2008; McGovern et al., 2003)	
Increase overall reliability by "producing systems that are more resistant to application and equipment failure and disruption" (McGovern et al., 2003)	
Upgrading a service and maintaining the system is cheaper than total application replacement (McGovern et al., 2003; Minoli, 2008)	
Parallel service development (Minoli, 2008)	

Table 2.2: Negative and positive characteristics of SOA

the services. Thus SOA requires a number of functionalities as an input, and any architectural solution that provides suitable information about the functionalities of a system could be used in collaboration with SOA. Even though SOA provide a logic for how to implement services according to standards, is dependent on other techniques to provide the essential and basic information about the required functionalities of a system.

Therefore, to continue the investigation for an EIS architectural process that could provide the basic and essential information about the functionalities of a system, the next section presents the results of analysing software architecture approach.

2.5.3 Software Architecture

This section presents a domain analysis of *Software Architecture*, to investigate its relation to EIS architecture.

Software architecture is components and the connectors between them (Rozanski & Woods, 2005; Clements, Kazman, & Klein, 2002). Software Architecture (SWA) “is the bridge between mission/business goals and a software-intensive system” (O’Brien et al., 2005, p. 1). Considering different definitions of software architecture, the terminology that this research is based on defines SWA as the sketch of the overall structure, showing “top-level design decisions, the interacting parts, their properties, and the main pathways of interactions” (Rozanski & Woods, 2005, p. 23). The main benefit of SWA, according to Rozanski and Woods (2005), is to create a better understanding of the system by providing an abstract and high-level design. However, techniques such as DODAF and SOA illustrates that software architecture can also start with a high-level and abstract design and lead to detailed design and an implementation phase. The review of EA solutions (i.e. SOA, DODAF, MODAF) presents that these solutions could be used as a strategy within an architecture to design an EIS. However, a global architectural solution such as software architecture is independent of a particular style or strategy. Therefore, to design an EIS architecture that is global to cover different EA solutions, we focus on the current process model of software architecture.

Figure 2.6 presents the details of a software architecture process. The first step is to consolidate the inputs, (Rozanski & Woods, 2005) defines the initial inputs as scope and context definition, and stakeholders concerns. From this information, the architect produces a number of scenarios in the

next phase. In the later phases (Step 5), an architectural option is evaluated against a number of requirements.

Bass et al. (2003) argue that the aim of an architecture is to support the quality attributes. Example of quality attributes are performance and time to market. Quality attributes are over and above the functionalities of a system, even though they are closely related (Bass et al., 2003).

The software architecture process that is suggested by (Bass et al., 2003) is based on and initiated from identifying quality attributes and defining scenarios that justify detailed information of the quality attributes. However, there is no standard source or process for identifying quality attributes.

The traditional software architecture process, including defining quality attributes, are purely based on requirements (Bass et al., 2003; Rozanski & Woods, 2005). (Clements & Bass, 2010a), in their recent work, argue that designing software architecture based on the requirements is not convenient for defining quality attributes as a generic concept that could capture various requirements. The concept of goals, on the other hand, is more high level and generic compare to requirements and could provide the initial information about the quality attributes. Goals are the architectural knowledge for today's industry and business-oriented systems (Clements & Bass, 2010a). The quality of quality attributes could improved by understanding business goals and using this knowledge to elicit quality attributes (Clements & Bass, 2010a). To achieve this objective they introduced a process model or method, Pedigreed Architecture eLicitation Method (PALM). The steps of this process are presented in detail in Section 8.2.2.

To review PALM, we applied it to the stroke care example. PALM steps are suitable for introducing a process model to a team of domain experts, to apply brain storming techniques to identify the goals and possible quality attributes. The steps are general and are based on identifying quality attributes from goals using *Scenarios*, nevertheless, no detailed instruction, method, or tool is provided to make this process model usable for non-PALM experts. In short, even though this method is aligned with our theory of using goal knowledge to design an EIS. architecture, there is no detailed guidance available on how to apply PALM to examples of EIS, in order to analyse this method and the results.

The lessons learnt from reviewing current software architecture solutions (i.e. PALM, (Rozanski & Woods, 2005)) is the basis for modelling a process that is aimed to design an EIS architecture, EAPM.

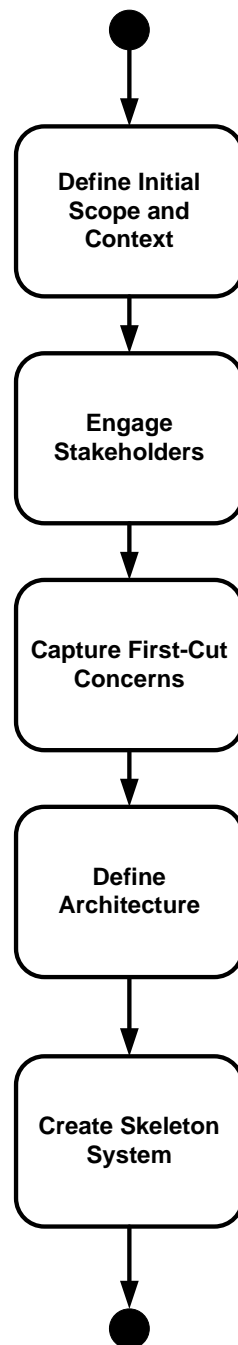


Figure 2.5: Main activities supporting architecture definition (Rozanski & Woods, 2005).

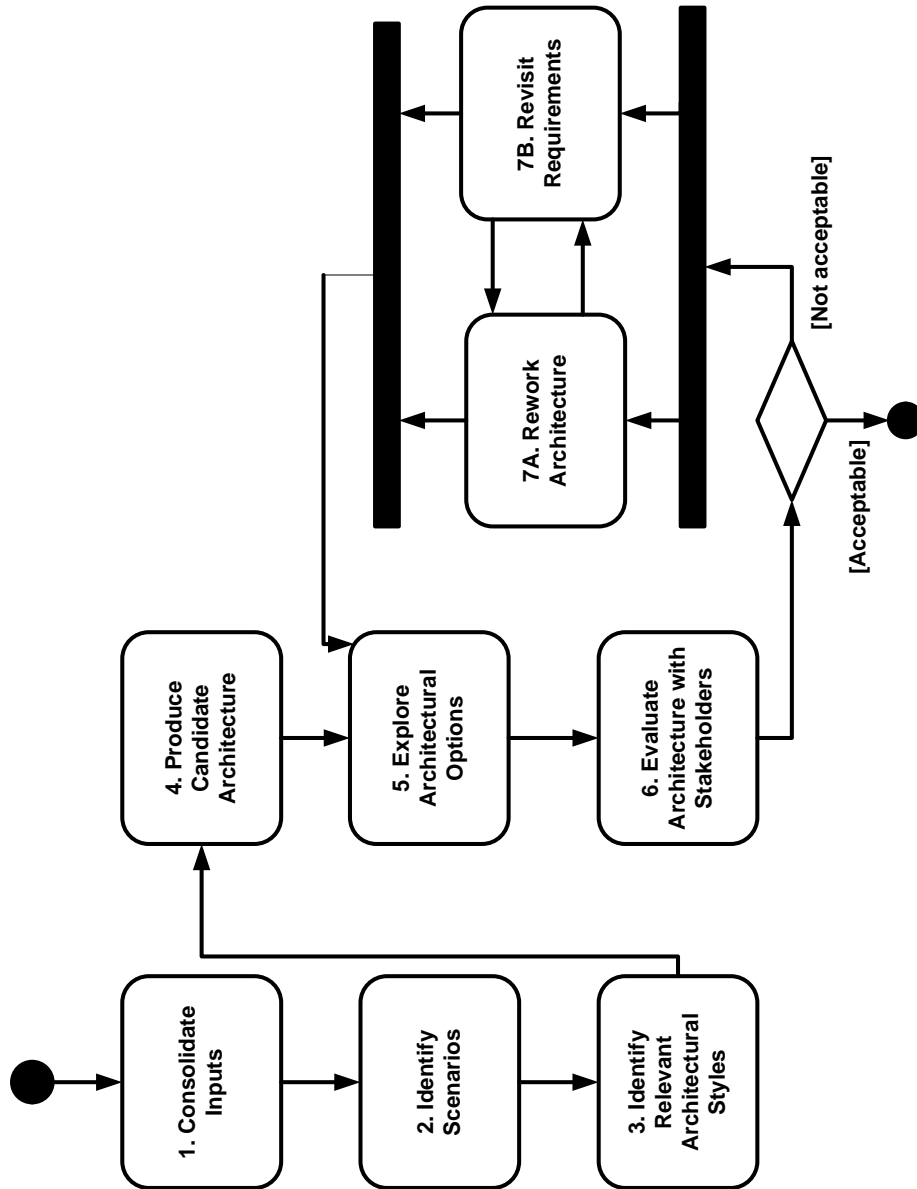


Figure 2.6: Software architecture process (Rozanski & Woods, 2005).

2.5.4 EIS Architecture Review Summary

The notion of EIS architecture has been reviewed from two perspectives: first, by appeal to EA concept and related technologies; and second, by appeal to software architecture solutions.

The analysis of EA technologies illustrates that the solutions (i.e. DODAF) could be used as architectural strategies and not an EIS architecture. An EIS architecture covers different aspects of an enterprise, in particular business processes. One way to address different aspects is to use the information collected from different view points (i.e. by using Zachman's Framework). Information collected from different point of view leads to further goal identification. Thus there is a traceability between the enterprise level goals and EIS architecture. This traceability is not explicit in any of the reviewed technologies so far. Therefore, to define an explicit traceability between enterprise goals and EIS architecture, a novel method should be developed. To develop and evaluate this novel method, this thesis uses two independent EIS case studies. One of these case studies (stroke care) also used during analysing different GOA. Section 2.7 introduces these case studies.

To define a transition from goals to EIS architecture. An approach to define a transition that is according to standards and could be reused by other practitioners, is to define a method that includes process models.

2.6 Definition Of Method

Because this thesis presents its contributions in the form of a *method*, this section presents its definition of method, including its constituent elements.

A method is composed of a process, a notation for expressing the outcome of the process, and underlying philosophy (Blum, 1994; Song, 1995; Chroust et al., 2010; Fuggetta, 2000).

The contributions of this thesis will be presented in the form of these components. In particular, we present: explicit process models to identify and structure EIS goals, and an explicit process model to trace and relate the influence of EIS goals to EIS architecture. The other method components (notation, philosophy) will also be presented.

2.7 Overview Of Case Study Domains

An empirical study is a means, not an end (Fuggetta, 2000); the method presented in the following chapters of this thesis do not present a final implementation solution of these specific EIS examples. The main objective is to

present hypothetical examples of EIS, based on real documents and realistic assumptions, to produce and illustrate the results of applying the processes developed in this thesis.

Two examples of EIS are used in this thesis and introduced in Section 2.7.1 and 2.7.2, stroke care EIS and Airport Crisis Management (ACM). Both of the examples are aligned with the EIS characteristics presented in Section 2.1.1. The main characteristic of both these case studies is that they address multiple business processes.

These examples are faced with general challenges of EIS, presented in Section 2.1.3; challenges such as design issues of a large scale and complex EIS (Berg, 2001), the design issues included the ones initiated from requirement engineering phase (i.e. defining unrealistic goals (Soumerai & Avery, 2010)) and the EIS architecture phase (Berg, 2001; Kaplan & Harris-Salamone, 2009).

In order to analyse different aspects of the two main case studies (stroke care and ACM), this thesis briefly reviewed experiences in developing EIS of the health care and crisis management domain. These examples of health care domain are London Ambulance Services's, patient electronic record in UK, and Denmark patient portal; and in crisis management domain, FBI crisis management information technology systems.

A review of general and specific examples of health care and crisis management information technology illustrates other common challenges for developing these EIS. These challenges are human challenges (e.g. lack of suitable stakeholders' participation, management issues, lack of clear ownership (Kaplan & Harris-Salamone, 2009; Rinkineva, 2004; Finkelstein, 1993)), technical challenges (e.g. lack of users' technical skills, poor development skills (Finkelstein, 1993; Kaplan & Harris-Salamone, 2009)), and challenges related to the characteristics of enterprises (e.g. complexity of business processes and changes within an enterprise including workflow and strategy changes (Kaplan & Harris-Salamone, 2009; ECRI Institute, 2011; Soumerai & Avery, 2010))

The development challenges of these examples, are predominantly those of requirements engineering and architecture design; this is the main motivation for choosing these two distinct examples of EIS. As a result, we use the first example, a hypothetical stroke care enterprise, during the development of the thesis method. The second example is used for an evaluation of the method, independent from the first example. The following two sections present stroke care and ACM examples, along with the lessons learnt from reviewing other examples of information technology from the same domains.

2.7.1 Stroke Care

Stroke is the third largest killer in countries such as US and UK (Schwamm et al., 2005; DH Stroke Policy, 2007) and a significant cause of morbidity and mortality. The effects of having a stroke for a patient could last for a life time and affect not just the patient but the family and carer. The cost of supporting stroke patients and their families also is high. One of the objectives of using information technology in the health care domain including stroke care is to reduce the cost of supporting patients (Soumerai & Avery, 2010)

Some of the main goals of using information technology in the healthcare domain, including stroke care, are to make smarter choices, have safer patients, reduce medical costs, improve workflow, improve quality of care, flexible enough to support organisations ranging from solo practitioners offices to national integrated delivery networks, maintain long standing beneficial patterns of communication, collaboration, and healthcare delivery (ECRI Institute, 2011; Soumerai & Avery, 2010; Shanmuganathan, 2010; Nagy, Simon, Sipos, & Kozmann, 1995; Schwamm, Audebert, Amarenco, Chumbler, Frankel, George, Gorelick, Horton, Kaste, Lackland, Levine, Meyer, Meyers, Patterson, Stranne, & White, 2009)

The collected processes of stroke care constitute an enterprise, and operate within the content of healthcare enterprise systems, the information system that support stroke care are designated EIS. A typical stroke care EIS is envisaged as consisting of three major group of information systems (Schwamm et al., 2009; Levine & Gorman, 1999; Anyanwu et al., 2003):

- Clinical information system (telestroke)
- Administrative information system
- External information system (i.e. information system for ambulance and secondary clinics)

Of these three areas most literature focuses on clinical information systems. Since 2001, new technologies have been implemented to support a stroke healthcare system. Telestroke (Schwamm et al., 2009) uses computer-based technology to integrate electronic medical information, clinical assessment tools, neuroradiology, laboratory data, and clinical pathways to bring state-of-the-art expert stroke care together (Levine & Gorman, 1999).

In the area of administrative information systems, a number of tools has been developed to support administrative and workflow systems (Anyanwu et al., 2003) to support stroke patients, their carers, health specialists, and healthcare administrative teams (Romano & Stafford, 2011).

A typical stroke care EIS is integrated with different systems within the whole healthcare enterprise. In the context of the stroke care EIS these systems might include support for patient records, emergency admissions, ambulance dispatch (Finkelstein, 1993). It is also possible that the stroke care EIS needs to interact with the systems outside the immediate healthcare enterprise, such as social services, long term rehabilitations (Anyanwu et al., 2003).

Integration with external system can be used to illustrate the importance of taking a holistic view of enterprise support. Failure to take a holistic view invariably leads to ineffective information system projects. Two well-known healthcare related projects demonstrate the issues. The London Ambulance Services's (LAS) fetching dispatch system (Finkelstein, 1993) shows the problems that can arise when a system is designed and implemented in isolation, ignoring different stakeholders needs and point of view in the process of identifying goals and requirements; whilst the patient electronic record (PER) system of Denmark (IBM, 2005) and UK (Morse, 2011) show holistic and isolated approaches to EIS.

LAS: Problems Of Isolated Information System Development

Finkelstein (1993) identifies four major groups of issue that leads to the LAS dispatch system failure in 1992. These issues could have been avoided by use of requirement elicitation techniques that considered the wider context of the enterprise, its strategy, and goals.

The first group of concerns technical issues such as lack of complete test and bugs in the code that leads to shortage of machine memory (Finkelstein, 1993). The second set of issues concern limitations in clear ownership and management. This led to lack of suitable communication between different parties and creation of unrealistic expectations (e.g. automatic improve of a number of existing working practices) that were not considered as goals or requirements in the requirement elicitation phase which led to lack of suitable planing (Finkelstein, 1993). The third group of issues concern evidence of misusing the LAS information system by some ambulance crew (Finkelstein, 1993). One assumption here is that this group of stakeholders was omitted from the early phases of requirement elicitation (Finkelstein, 1993). The fourth group of issues relate to over ambitious and unrealistic top-level expectations of the system (e.g. an expectation from healthcare management that LAS information system would automatically improve a number of existing working practices). The LAS report evidence illustrates that the

implementation was according to the design, however the design has failings that are due to the unsuitable requirements and undefined top-level goals. One of the solutions (suggested by Finkelstein (1993, p. 8)) would be to have shared and agreed objective between IT managers and organisational managers.

The investigation of reasons for LAS information system failure illustrates the need for addressing requirement issues and identifying top-level goals that are based on the communication and shared understanding between different parties is part of the solutions. It is also evident that traceability from high level enterprise goals to requirements and strategic design (i.e. architecture design) is important for such evolving and emergent healthcare information systems (Weber-Jahnke & Onabajo, 2009).

PER: Examples Of Successful And Unsuccessful Integration

PER is another example of an external information system that would be relevant to the stroke care EIS. It also illustrates the importance of requirements and traceability to design as part of a successful information system. The aim of the PER information system is to “reduce reliance on paper files, make accurate patient records available at all times, and enable the rapid transmission of information between different parts” of a national healthcare system (Morse, 2011, p. 4).

The Denmark PER information system is a successful case that IBM has not published public documents about the development process and challenges. In Denmark healthcare system development scenarios are used to identify the top goals and expected functionalities of the system-to-be and to present them to different groups of stakeholders (IBM, 2005). Other contributing reasons to the success of PER in Denmark is the smaller population and the higher IT awareness of the enterprise, IT savvy compared to the UK NHS (Harrell, 2009).

On the other hand, the UK PER information system is suffering from many development issues, leading to changes in the plan from its initial stage (Morse, 2011). Some of the identified issues are the lack of shared understanding and lack of clarity between departments and IT suppliers; changes of requirements; and developing parts of the information system in isolation (Morse, 2011). Lack of communications and changes in the goals and requirements within UK healthcare’s large number of stakeholders, become the motivation to develop the PER information

system locally and according to set of standards and not in enterprise level (Morse, 2011).

Considering the top goal of PER, which is to provide rapid transmission of information between different parts of the health centre, the isolation of parts of PER information system develops a conflict with a top-level enterprise goal. To conclude, a top-level goal of PER is to improve services and quality of patient care, however the development of this information system has fallen below expectations (Morse, 2011). One of the IT sectors argues the changes in the requirements leads to difficulties in the development (Morse, 2011). In addition to the requirement changes Soumerai and Avery (2010) lists the following issues: unrealistic goals, lack of suitable technical support, different vendors developing different part of the system, incompatible products to build integrated health system, software error and unreliable technology, hidden implementation and no public documents on development phases. One of the Soumerai and Avery (2010) suggests to build a successful healthcare information system is to use open source resources and materials that a larger number of people could collaborate in its improvement, testing, and evaluation.

Stroke Care EIS: The Enterprise Context

The review of these two healthcare information systems highlights the importance of requirement elicitation solutions such as GOA and tracing them to strategic design in developing a healthcare EIS. The aim of stroke care EIS is to support stroke care functionalities to improve stroke prevention, treatment, and rehabilitation. Prevention aims at helping high-risk populations to reduce the chance of having a stroke by changing their lifestyle and taking certain precautions (strokeprevention.org, 2011). In US, only 76 percent of annual stroke patients have first time stroke, so it is important to provide support for this phase to increase the awareness of first stroke and recurrent stroke (Gorelic, 2009). The main goals of prevention phase is to secure the privacy of the visitors who are attending the programs, not to reveal their information. Prevention aspect with the main goal of secure visitors data is not the focus of this research.

The second aspect of stroke care is treatment during the first 72 hours of stroke. The goal of this phase is to treat patients to prevent long-term disability and save lives (NHS, 2011). The functionalities and goals of this phase is more focused on medical and clinical goals however,

to demonstrate an EIS with enterprise strategic level goals, part of rehabilitation phase is considered for applying process models. The main reasons for this choice are as follows (National Institute for Health and Clinical Excellence, 2008):

- Even though it has medical treatment goals, it includes a fair number of strategic enterprise level goals
- It includes different groups and departments such as health specialists, patient, carer
- It includes different services such as treatment after stroke, support of patient and carer, diagnosis for recurrent stroke, patient transport system
- It includes short term goals and plans such as first two weeks after stroke
- It includes long term goals and policy to support a life time of patients who suffer from stroke and their carer

A main challenge of implementing the rehabilitation phase is lack of agreement between health professionals. However, as this thesis does not address the implementation of an EIS for a particular region with particular policy, at this stage we assume the health professional are agree to communicate with each other and share their knowledge and opinion to improve the patients' health. This example also is limited to a number of high-level goals and does not address all the goals and expectations of a rehabilitation phase.

Figure 2.7 presents the high-level view of a hypothetical stroke EIS in relation to parent EIS such as healthcare system. In this figure, a healthcare system is a wider enterprise, and the stroke care is a child enterprise. Within stroke care, there are three child enterprises: stroke prevention, treatment, and rehabilitation. Each of these enterprises are in collaborations with other parts of health system such as ambulance system and organisations independent to health system such as private physiotherapists, which are not presented in this figure.

A stroke rehabilitation aspect of stroke care is an EIS because it has the EIS characteristics demonstrated as follows:

- Has different business processes such as the ones addressing stroke prevention, the ones addressing stroke treatment, and the ones addressing stroke rehabilitations

- Supports organisational goals such as improving stroke care and providing applications to improve stroke care
- Involves and orchestrate multiple business processes such as support the treatment of a patient and provide a secure environment for the patient's data
- Includes multiple partners such as laboratory system, ambulance system, and private physiotherapists
- Has evolutionary development, to minimise the risk of using such systems in real environment when all the elements are not fully tested (lesson learnt from (Finkelstein, 1993))
- Has a dynamic architecture to deal with the changes in the health policy or extending the system to new applications and regions
- Has geographical distribution to support the patients in different parts of a country or region including rural area
- Contains sensitive health related data (i.e patients electronic records)
- Support the changes in technology, health science, policy, and business processes
- It is an open system as it interact with human (i.e. health specialists, patients, ambulance staff) and hardware devices (i.e. CT scan and MRI equipment)

As mentioned earlier, this thesis will focus on the rehabilitation phase.

In conclusion part of an iterative process of goal identification and tracing it to the strategic level of EIS architecture for a hypothetical rehabilitation phase of an stroke care is presented in Chapter 6. To address the requirements of emergent and evolving nature of healthcare information system, suitable traceability to and from a goal model to different phases of development process is required (Weber-Jahnke & Onabajo, 2009). Therefore, chapter 6 also presents a demonstration of tracing part of the stroke EIS rehabilitation goals to EIS strategic level using the methods developed in this thesis in Chapter 3.

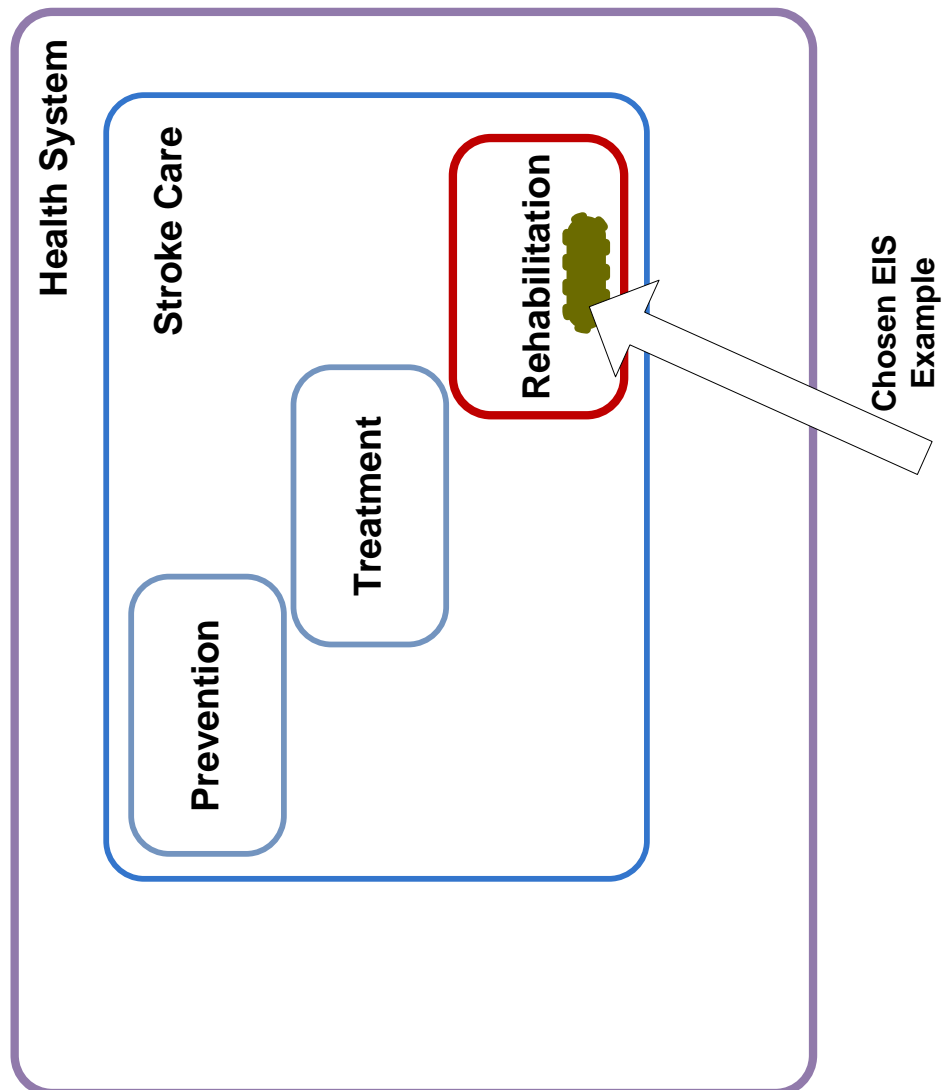


Figure 2.7: The structure of the parent goal and its three child goals.

2.7.2 Airport Crisis Management

This section introduces the second EIS example, Airport Crisis Management (ACM). In addition, to demonstrate why this example is suitable for this thesis, a number of EIS characteristics that match with ACM characteristics are reviewed. To gather more information about ACM characteristics and development of similar information systems. This section also briefly reviews the Federal Bureau of Investigation (FBI) Crisis Management System (CMS).

FBI Crisis Management: Example Of Successful CMS

“Crisis management involves the activities of a great number of agents confronting the same problems” (Rinkineva, 2004). CMS provide the ability for stakeholders to make timely and appropriate decisions that translate into decisive actions (Neubauer, 2007). Crisis management is an enterprise, because “no single entity can be responsible for the entire management of an enterprise” (Iannella, Robinson, & Rinta-Koski, 2007). The main goals for a CMS is to create an interoperable network of coordinated agents, to share the information and knowledge in a suitable time period (Rinkineva, 2004). To achieve this goal a CMS requires applications for communications, applications for knowledge sharing, agreed policy, and implementation standards (Rinkineva, 2004).

Other than the technical issues in developing such an EIS that is discussed at the beginning of Section 2.7, some of the main issues faced by the people who use this system is human competition rather than coordination to share the resources and information (Rinkineva, 2004).

This thesis identified several commercial crisis management software (EmerGeo, 2011; ERMS, 2011); a good example of such system is developed for FBI. FBI CMS example is for a brief review because FBI CMS, Orion, is an example of working and successful EIS for FBI (Critical Incident Response Group, 2007), even though there are still investigates about different approaches to share the information in a more secure and accessible environment (Neubauer, 2007). Orion provides case management and related information processing capabilities to support federal, state, local, and tribal law enforcement agencies during a coordinated response to a variety of events and critical incidents. Orion has successes in information processing such as capturing data provided in thousands of e-mails and phone tips and to convert raw

information into actionable intelligence. However, there is no evidence about development methodology, that can be reviewed further in this thesis. One of the main challenges of developing a CMS such as Orion is to define a cooperated success measure and desire in an uncertain environment (Critical Incident Response Group, 2007). This challenge highlight the difficulties in identifying and structuring CMS goals from different views and defining the success measures that are aligned with the goals.

ACM: The Enterprise Context

“Crises are without doubt headline grabbers, and, it seems in recent times, there have been many high profile events, which have hit the commercial aviation sector hard. Global economic crisis aside, one of the most disruptive and costly events to have hit the industry in recent years has been the volcanic ash cloud which had loomed over European Airspace. For such an unforeseen act of Mother Nature to have such a devastating affect on Europe’s airports and airlines (cost to airlines 200m Euro a day according to IATA) is unprecedented and has once again put emergency planning experts to the test around the world” (ACM, 2010).

“Crisis Management immediately brings to mind the rare occurrence of aircraft accidents but there are many more challenges facing airports and airlines which need contingency planning. Implementing appropriate measures to cover any crisis, which can lead to delays, damage to property or customer safety & well-being will reduce the adverse affect upon all stakeholders. Both airports and airlines rely upon public confidence that their operations are running in secure, safe environments” (International Airport Review, 2009).

An analysis of ACM reveals the following challenges in developing these systems (Magister Ludia Aviation and Softsolutions, 2011):

- Late activation of procedure
- Late communications of relevant information
- Late communications to relevant people
- Information overlapping
- Incorrect application of procedures
- Incorrect storage of information

- Only minimal training for the people involved

For this thesis an example of ACM is used that is based on a real development of an ACM, which used the state of art techniques and is developed in a European airport (MODELPLEX Consortium, 2007). The real case is commercial and the data is confidential.

This example of ACM is an EIS because it is aligned with the following EIS characteristics, presented in Section 2.1.1.

- Support multi business processes such as air traffic control tower (ATCT) business processes, emergency assistance within flight service station (FSS), and air route traffic control centre (ARTCC), which control the flight from outside an airport (NATCA, 2011).
- Support organisational goals such as implement appropriate system to cover any crisis, which can lead to delays, damages to properties, or customer safety and well-being
- Includes multi partners such as within airport grounds (i.e. Police, Firemen, ATC Tower, Airline Companies, Security Companies) and external to airport grounds (i.e. Hospitals, Police, Media) (NATCA, 2011)
- Has evolutionary development to minimise the risk of using such systems in real environment when all the elements are not fully tested
- Has dynamic architecture to deal with the changes in the ACM policy or extending the system
- Has geographically distribution to within and outside airport (NATCA, 2011)
- Contains sensitive airport and travellers related data (i.e. reasons for a certain system failure)
- Interacts with other systems (i.e. Human, Radar Approach Control)

Even though the example is based on a real ACM case, the aim of this thesis is not to implement a solution for a real case of ACM. To illustrate this thesis method, a number of realistic scenarios are generated.

So far, a review of the enterprise, related technologies, method, and case studies have been presented. These reviews lead to identifying a number of gaps that are the basis for research questions and hypothesis, presented in the following sections.

2.8 Hypothesis

The literature review shows us that there is a:

- Lack of explicit process models to identify and structure EIS goals
- Lack of explicit process models to trace and relate the influence of EIS goals on EIS architecture

This leads us to the following hypothesis. The hypothesis of this thesis is based on a number of research questions, presented in the introduction chapter, that guide the research exploratory path. The objective of this section is to present a testable hypothesis that is aligned with the research questions and addresses the gaps and limitations presented in the Section 2.8.

Hypothesis:

Process models can be developed to provide a precise, repeatable, and documented set of structures for *identifying* and *specifying* EIS goals, and for *relating* EIS goals with a strategic-level enterprise architecture.

The critical terminologies that are used to build this hypothesis are process model (defined at Section 2.4), goals (defined at Section 2.2), strategic-level EIS architecture (defined at Section 2.5.2).

A hypothesis is a testable statement, therefore, the testable features and keywords of this hypothesis are: precise, repeatable, and documented process models.

The goal of this thesis is to define a method including process models that could be evaluated in line with the hypothesis's testable criteria in addition to delivering the functionalities defined by the hypothesis, identifying, specifying, relating. These process models are for identifying, structuring EIS goals and reflecting on EIS architecture.

The sub goal here is to define the testable criteria and how they could be addressed in this thesis.

Precise: to define a precise process model, we used the SPEM process structure metamodel (Pereira et al., 2011) and address the well-formedness rules using EPF.

Repeatable: To define a process model that is repeatable and used by other practitioners, we document the process and applied them to independent examples of EIS. In addition, this explicit process model has a static web-based tool that could be accessed in different platforms.

Documented: a process model could be documented using natural language and graphical notations (see Section 2.4). To document the process models of this thesis, in addition to natural language and graphical notations an eclipse plug in (EPF) is used to build supporting tools based on the SPEM standard.

Documenting a process model and developing supporting tools helps to use the process model by the domain experts and not only the IT technical experts.

2.9 Conclusion

In summary, Chapter 2 presents a literature review that is motivated by the top level research questions posed in Chapter 1. To investigate the answers for the first research question that is focused on the essential characteristics of an EIS, Section 2.1 presents an analysis of the enterprise and EIS concepts. This analysis includes a novel definition for EIS, its characteristics, its challenges, and EIS examples. The main lesson learnt at this phase is that to follow a systematic research, the first step is to clarify the main concepts used as the backbone of the research. Some terms such as EIS are used frequently in industry and academia, yet, the term and concept suffers from an unclear definition.

The research question that focused on the reasons of failure in delivering the functionalities and addressing stakeholders' goals, inspired a further analysis of requirement engineering approaches, in particular GOA, and design phases, in particular architecture design. Therefore, Sections 2.2, 2.3 and 2.5 present an analysis of the current approaches in these domains and presents a critical review. The main learning outcome of this analysis and critical review is that the GOA and architectural design solutions strongly dependent on the experience of their expertise. This becomes a main challenge for developing a method that aim to develop a collaborative environment for main groups of EIS stakeholders with little experience of GOA and EIS architecture.

The final research question focuses on a knowledge that could help to identify essential functionalities for an EIS. This research question motivates a further analysis of a method that uses the results of analysing GOA and EIS architecture design solutions, and not only be a roadmap for identifying and structuring goals of an enterprise, but also trace those goals to EIS architecture design. The main lesson learnt from investigating this research question is identifying the elements of a method in software engineering domain (Section 2.6). One of the essential elements of a method is a process

model. The importance of defining a process model (Section 2.4) and not a specific process to deal with flexible requirements of an EIS is one of the main lessons learnt during analysing process and process model's requirements.

This thesis integrates multiple areas in software engineering domain, including, requirement engineering, software and EIS architecture, method and process modelling. All these areas of software engineering have a common characteristic, that is qualitative results rather than quantitative. This develops challenges in proposing solutions that could formally prove to address the problems. Therefore, during this thesis, mainly evaluation chapter, qualitative techniques are used to support the results that are developed during this thesis. These results have several limitations, including being developed by one person and applied and evaluated by the same person. Nevertheless, it is a recognised limitation of many PhD thesis.

Next, we commence presentation of the method developed in this thesis. The first step is to present the novel process models that enable identification of enterprise goals and tracing thereof to a strategic EIS architecture. These process models – KAOS- β and EAPM – are presented in Chapter 3.

Chapter 3

Method: Process Model - From EIS Goals to Strategic-level Architecture

This chapter commence the presentation of the new method developed in this thesis, a method for systematising the early stages of development of an EIS. In particular, the method is for eliciting, identifying and capturing goals, and refining and connecting those goals to an early EIS architecture. The method addresses a gap identified in Chapter 2, lack of systematic consideration of the diverse and volatile interests and objectives of enterprise stakeholders, and the influence these have on the identification of goals for the EIS architecture design.

The concept of a method was introduced in Section 2.6 and the context of this thesis method is shown in Figure 3.1. The method comprises two processes, KAOS- β (derived from van Lamsweerde (2009) KAOS) and EAPM (motivated by enterprise architecture and software architecture solutions). The method assumes existing information about enterprise strategy (i.e. reports of structure of enterprise, feedback documents, interviews, enterprise/system policy, review of similar information systems, report of current systems, requirement documents) on the current structure, objectives and stakeholders of the enterprise. The result of applying KAOS- β and EAPM is a strategic architecture for the information systems to support an enterprise.

The method consists of the two aforementioned process, KAOS- β and EAPM, which are connected as indicated in Figure 3.1. The inputs to the method consist of enterprise strategy documents. KAOS- β generates a structured set of goals, which are used as input to EAPM. The output of EAPM is strategic architectural information. The combination of KAOS- β and EAPM systematises the process of transforming strategy documents to strategic ar-

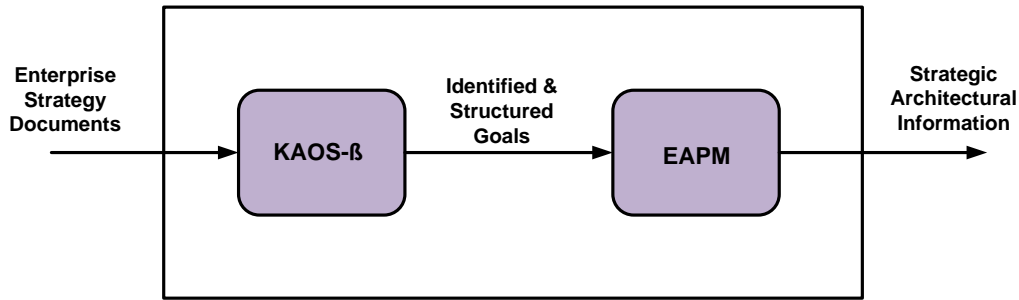


Figure 3.1: Method to address the gap in enterprise level systems analysis: the context of KAOS- β and EAPM.

chitectural information. By connecting these two processes, we are able to trace early-stage EIS architecture decisions to enterprise-level goals, and ultimately strategies.

The rest of this chapter is organised as follows: Section 3.1 presents KAOS- β process model, its elements and structure. Section 3.2 introduces EAPM, its elements and structure and Section 3.3 summarises the content of this chapter.

3.1 KAOS- β : A process for analysing the strategic goals of an enterprise

One of the gaps identified in Section 2.8 is the lack of explicit process model to identify and structure EIS goals. The objective of this section is to introduce KAOS- β elements and structure to address this gap. KAOS- β is a process model used to identify and structure the goals of EIS by analysing the enterprise strategy documents.

3.1.1 KAOS- β Elements

The main elements of KAOS- β are enterprise strategy documents, goals, modules, and agents. These elements and their relationships are presented in Figure 3.2 using a metamodel.

A number of reliable and standard enterprise strategy documents are the core of KAOS- β . Analysing these documents leads to identifying and defining a number of goals and modules. Each module could suggest one or more goal

and each goal could belong to one or more module. Further analysis of the goals leads to identifying more goals and identifying negative affects of goals on each other (i.e. conflicts, threats, obstacles). Goals are addressed by zero or more agents allocated to them. Each agent addresses at least one goal.

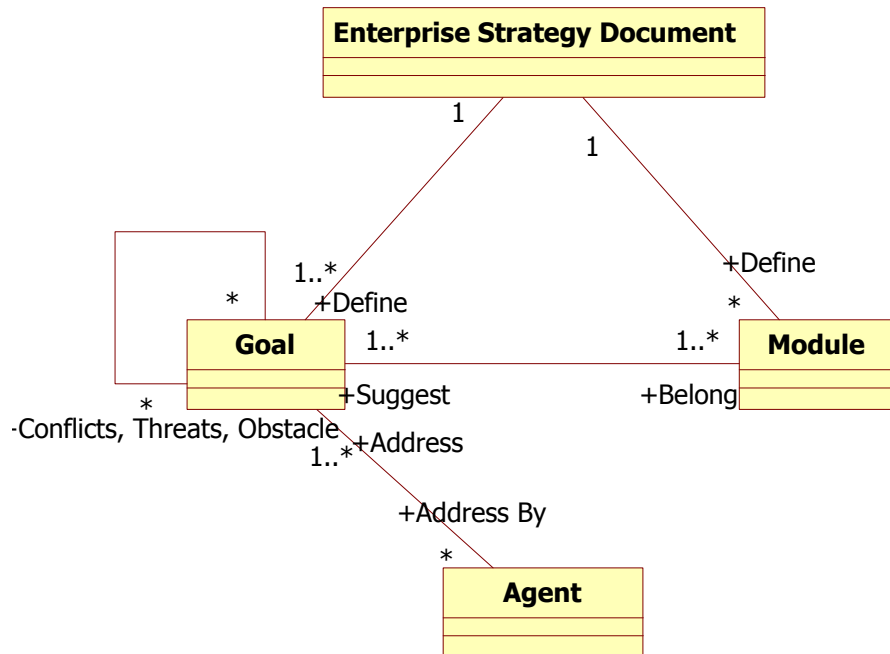


Figure 3.2: KAOS- β Elements.

3.1.2 KAOS- β Structure

Figure 3.3 summarises the structure of KAOS- β process model. In KAOS, *task 1* concerns identification of the strategic level goals. For an EIS, this involves searching for strategic objectives, business goals, domain-specific objectives, and goals that could be refined by analysing requirements and problems. To identify goals, designers could analyse the statements about an EIS by asking what goal(s) does each statement exemplify? what goal(s) does each statement block or obstruct? (Antón, 1996). In addition, analysing and searching for action oriented or objective oriented keywords (van Lamswerde, 2009; Antón, 1996) has been useful to identify some of goals in practice. For our EIS example, we benefit from defining and analysing scenarios to visualise the EIS responsibilities and expectations; the benefits of using scenarios is also mentioned in relation to GBRAM (Antón, 1996).

In *task 2*, where an enterprise has many goals from different view points, the designers use modularity to structure and manage the goal model. The heuristic used to derive modules in the EIS example is to consider participant groups, many of whom are later designated as agents. Modularity is a critical step: from this point, each KAOS- β task is applied to each module. Modularity, which is suggested by GSN (Kelly, 2001), is an approach to handle the complexity of large number and diverse goals.

Within each module, *task 3* is akin to goal refinement in KAOS and GBRAM. In *task 4*, the designers document the goals, which lead to identification of missing, redundant, or mis-specified goals: the documentation process is thus part of the validation of the identified goals. Iteration of tasks 3 and 4 is usually required. Table 3.1 presents a template for documenting EIS goals. Some of the elements of this table are taken directly from KAOS, such as goal name and *def*. The starred elements are required to provide the minimum information for stroke care EIS example to trace the goals to the source and present them to a broader audience using scenarios. The **priority** criterion could be defined in different ways, such as high, low, medium, or using a numbering system. As for **success measure**, because KAOS- β does not aim to lead directly to a system implementation, success measure is not a mandatory criterion, and it is difficult if not impossible to define it at a strategic level, but where success measure do exist, it can be documented in the template. This criterion and the scenario criterion are added to the template in the later iterations of using and analysing KAOS- β ; they are useful mainly in extending KAOS- β to EIS architecture.

In *tasks 5 and 6*, the designers refine goals and document the refinement links, creating traceability within the goal structure. Iteration occurs as new goals are identified. Table 3.2 presents a filled template for documenting the

refinement links. It is filled because we find an example that clarifies the content of the template. In this template, **ID** is an identifier, **name** is a descriptive name for the link, **SysRef** is the state of the system (i.e. system-as-is, system-to-be, version of a system, or sub-system), **Status** is the condition of the refinement process (i.e. ready for further goal refinement, or ready to transfer to requirements by adding implementation data), and **tactic** is the source of refinement strategy (i.e. designer assumption, system documentation, policy).

In *Task 7* agents are identified similar to KAOS, it includes human and non human agents. The main difference is that in KAOS- β , more than one agent could be associated with each goal. In *Task 8* the designers associate the agents to goals similar to KAOS; no particular information is recorded at this stage. In *Task 9* the designers search for obstacles, threats, and conflicts, with the appropriate iteration to step 3 to check for any subsequent changes in the goal set or the links. These iterations are part of evaluating and improving the goal structure.

ID *	
Name *	
Def *	
Scenario *	
Context	
Priority	
Source *	
Success Measure	
Issue/Notes	

Table 3.1: Template for the structured documentation of KAOS- β 's goals. The criteria with the star sign are strongly recommended to be filled.

ID	ACML1-2
Name	Link from ACMG1 to ACMG2
SysRef	Sys-to-be
Status	Goal under further refinement
Tactic	Designer's assumption is based on document ¹

Table 3.2: Example of the structured documentation of KAOS- β 's links refinement.

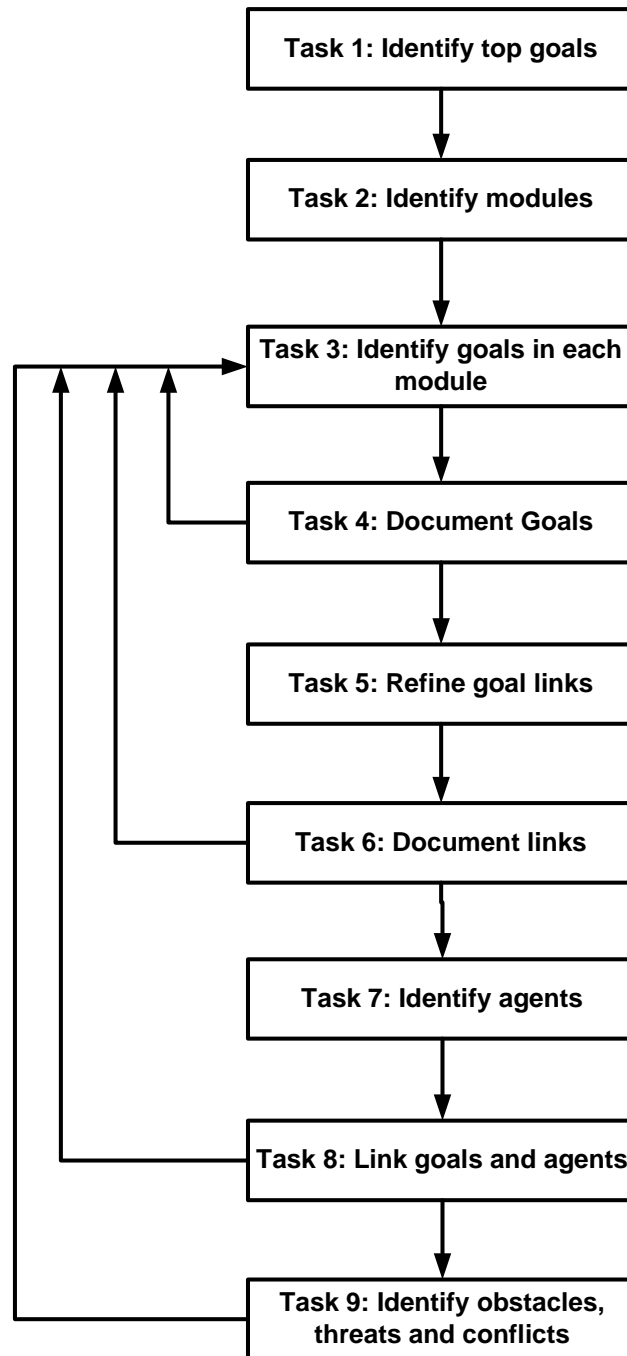


Figure 3.3: Activities of the KAOS- β Process (Original KAOS activities and heuristics are presented in Figure 2.3).

3.2 EAPM: A Process Model to Reflect Goals in Strategic-Level Enterprise Architecture

The second gap identified in Section 2.8 is the lack of explicit process model to trace and relate EIS goals to EIS architecture. The objective of this section is to introduce a new process model, Enterprise Information System Architecture Process Model (EAPM), to address this gap. EAPM is proposed because, as mentioned in Chapter 2, there is a need to design an EIS architecture based on goals rather than requirements; existing architecture methods are targeted to addressing single system software and data architecture or to documenting different points of view rather than designing a strategic-level EIS architecture.

EAPM can be used to derive architecture requirements from a set of goals (such as those developed using KAOS- β). The elements of EAPM are based on enterprise architecture and software architecture solutions presented in Chapter 2. The documentation notation is motivated by ATAM (Clements et al., 2002) and the underlying philosophy is that of software architecture presented by (Rozanski & Woods, 2005; Bass et al., 2003; Clements et al., 2002). The elements of EAPM are introduced in Section 3.2.1 followed by EAPM structure presented at Section 3.2.2.

3.2.1 EAPM Elements

This section describes the EAPM elements and is followed by the description of the sequence of tasks in the next section.

The main elements of EAPM are Goals, Quality Attributes, and Strategies. Figure 3.4 presents the relationship between these elements and other related elements, using UML syntax. As can be seen, on the top, outside the dashed box, a **system** is associated with **goals**, **quality attributes**, **architectural description**. A **system** provides the source of a number of goals and quality attributes. An architecture description could be evaluated against the system documentation.

Each **goal** could generate several **quality attributes** and similarly any **quality attributes** could be justified by one or more **goals**; this presents the association between goals and quality attributes.

Measurable values are used, when possible, to evaluate a strategy. Hence, **Strategy** uses a measure to evaluate the architecture. The information for **Measures** could be generated by goals, quality attributes, standard specifications and limitations of the system, and architect's expertise.

Strategy suggests one or more **styles** and consequently **architecture descrip-**

tions. Examples of strategies in the domain of EIS architecture are SOA and DODAF; examples of styles are client server and three-tier. Strategies could have several styles for their implementation. Hence, **style** could associate with other, more detailed **styles**. **Style** or combination of styles suggest one or more **architectural description**; hence each **architectural description** contains one or more **style**.

To design an architecture for an EIS system, the goals and quality attributes could provide information required to generate measurement values to help designers to evaluate the results. Based on the description of quality attributes a number of strategies would be nominated to address them. A combination of strategies could create a style or a set of styles of architecture for the EIS system. Section 3.2.2 presents the EAPM process structure that is based on the elements and linkage shown in Figure 3.4.

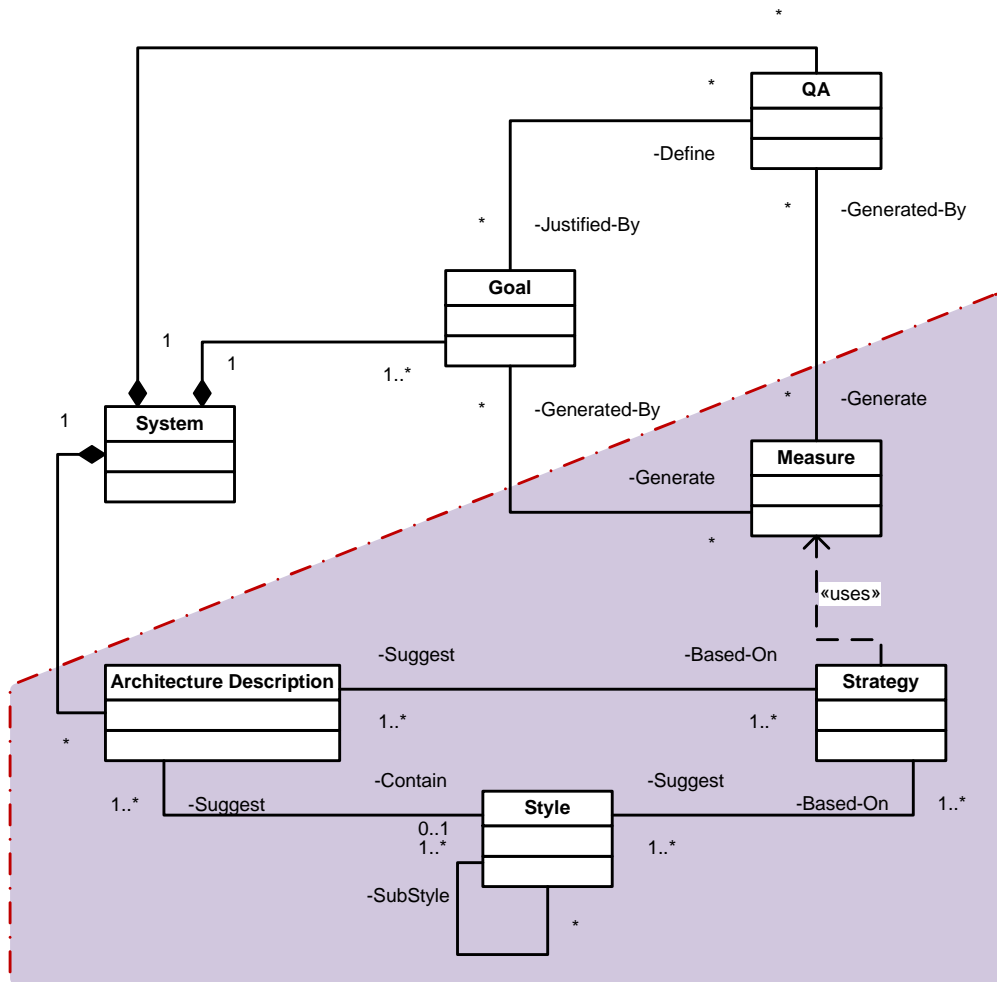


Figure 3.4: The elements of EAPM presented as a metamodel. The dashed area present the components and connectors that provide detailed information for the choices of architecture.

3.2.2 EAPM Structure

The structure of EAPM uses the elements from Figure 3.4 and organises them in sequential tasks. There is iterations between tasks to either collect more information or revise the elements such as quality attribute and strategy.

Considering the elements of EAPM that are presented in Figure 3.4, this section presents the structure of EAPM. This structure uses the elements and organise them in a format of sequential tasks. Even though the tasks are sequential, where required, there are iterations between tasks either to collect more information or to revise the current one. The structure of EAPM is presented in Figure 3.5. Following is the description of each task.

Task 1: this task establishes a basic and essential knowledge to initiate EAPM. This task is based on an activity introduced by (Rozanski & Woods, 2005). For an EIS architecture the input data include, but are not limited to, goals, agents (actors), links, questions, indirect stakeholders. These maybe provided as KAOS- β outputs.

Analysing the input data, including the goals, provides knowledge to determine the quality attributes and drivers. Hence the next step is:

Task 2: this task is based on architectural guidance given by (Bass et al., 2003), to define less than ten quality attributes in the first iteration, to keep the size and complexity of the design in control. To identify quality attributes, EAPM uses goal description such as that provided by KAOS- β and defines one or more scenarios to support that goal description. Analysing the goal description and related scenario triggers identification of candidate quality attributes that capture the essence of the goal. For example, a health-care EIS (Harrell, 2009) goal could be the easy and secure access to personal health-care information. This goal indicates the quality attributes of security and usability (user friendly interface).

Based on the information collected from the system, standards, specification, and the output of KAOS- β , designers could identify measures or measurable values. These measures could help the designers and the group who evaluate the results to have a set of standards to measure the success of the results. The next step accordingly is:

Task 3: based on the information collected from the system, standards, specification, and the output of KAOS- β , designers could identify measures or measurable values. These measures could help the designers and the group who evaluate the results to have a set of standards to measure the success of the results.

This task is motivate by evaluation activities in (Rozanski & Woods, 2005) and ATAM (Clements et al., 2002). Similar to goals, if possible, designers should define measure of success for quality attributes to evaluate

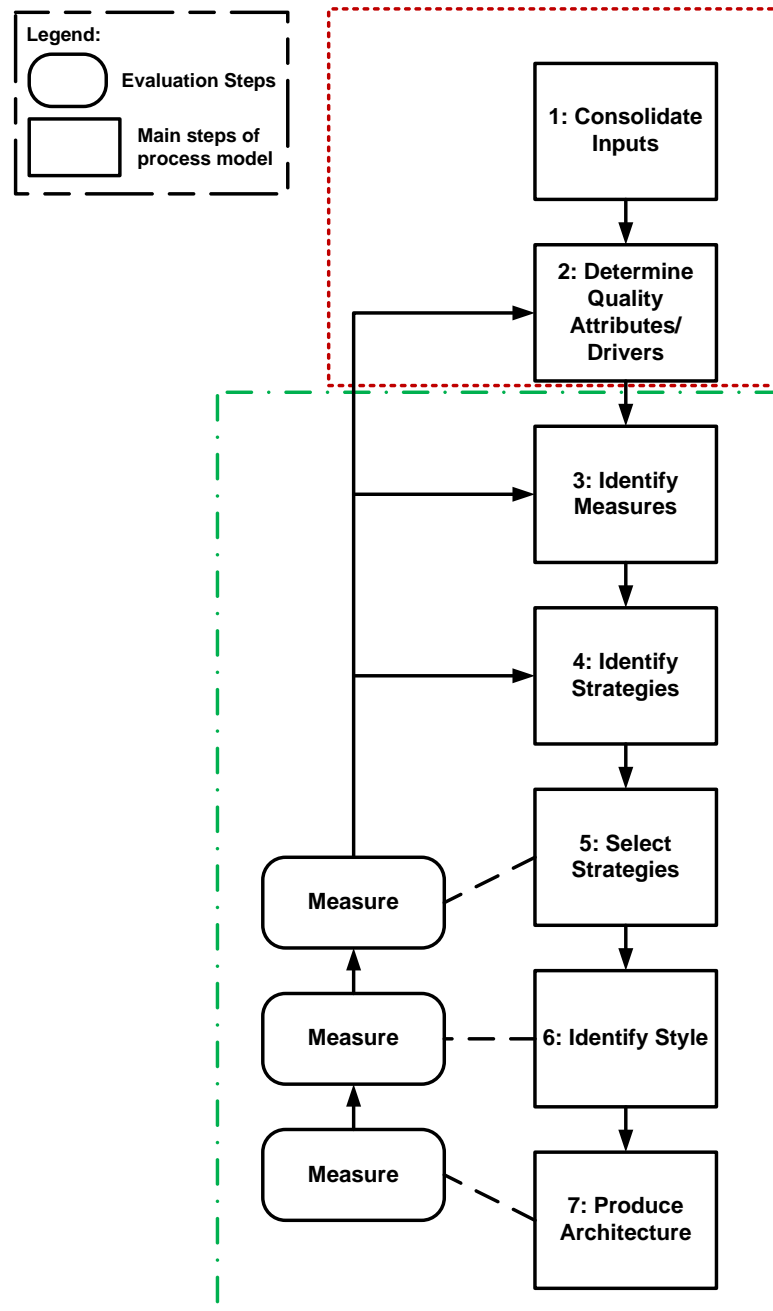


Figure 3.5: The EAPM. The green (dashed) box is a support for the red (dotted) box.

how a quality attribute is succeed. The measure of success could be defined loosely at this level, where defining quantitative measures could not be valid. Same as quality attributes, the goals and their scenarios can be used to investigate the measures of success; this could be a number (i.e. supporting more than 100 members of staff), or a qualitative measure (i.e. suitable for elderly patients).

Task 4: this task focuses on the identifying possible strategies and not the best one. The task is motivated by evolving architectural strategies to match with business strategy (Nedstam & Staples, 2007). Current EA solutions (i.e. SOA, DODAF) indicate that they could be used as a strategy within the EAPM. Detailed strategies such as two factor authentications (Wikipedia, 2011g) or physical access control to satisfy security quality attribute also could be identified in this stage.

Strategies could have conflicts with each other, hence after identifying them the next task is to select one or more suitable strategies.

Task 5: each combination of strategies introduces different architectures. In an ideal case, designers create different architectures and compare them to find the better option. In Figure 3.5, the **select strategies** box addresses this activity. The measurable values that were identified in step 3 could affect how to select combination of strategies, and in future phases how to identify style and evaluate the produced architecture.

To implement a strategy, a number of styles could be used.

Task 6: choosing a strategy and a style are interrelated steps. This task is motivated by an activity introduced by (Rozanski & Woods, 2005). Architectural style (i.e. client-server, pipes-filters) is a vocabulary of components, connectors, and constraints that could be used in an instance of an architecture (Garlan & Shaw, 1994). Architectural styles are linked to architectural strategies.

Task 7: this task is motivated by an activity introduced by (Rozanski & Woods, 2005). This final stage combines all the information of the previous tasks and present one or more architecture description and diagram.

Note that if the identified measure of success in task 5, 6, or 7 changes, it might reflect and change the previous tasks. The **measure** boxes in Figure 3.5 address this point.

3.3 Conclusion

A process or a process model is an element of a method (Section 2.6) and this chapter presents the first step towards defining a novel method by developing two process models: KAOS- β and EAPM. The result of using these process

models is systematic identification and structuring a number of goals that are traced to strategic EIS architecture.

Several lessons have been learned during the process of analysing and defining KAOS- β and EAPM including there is no one approach and solution for identifying goals; different materials and approaches could be used to identify suitable goals. In addition, the process models developed in this chapter could amend based on the specific requirements of each project. A tailored process for each project could capture different view points of stakeholders more accurately.

To capture different views of diverse stakeholders, we identified scenarios as a suitable solution to create a shared platform for identifying and justifying goals. Another lesson learnt during defining scenarios, is the benefit of having domain experts and IT experts. One of the limitations of this thesis is the lack of required access to these stakeholders. Thus we believe the participation of these stakeholders leads to identifying and structuring goals that reflects the true goals of an enterprise's functionalities and avoid defining goals that leads to unrealistic expectations from development team.

Chapter 3 presented the KAOS- β and EAPM. Continuing in our overall presentation of the method, the next chapter presents the method's underpinning philosophy.

Chapter 4

Method: Philosophy

The philosophy of science is “the critical study of the basic principles and concepts of a particular branch of knowledge, especially with a view to improving or reconstituting them” (Dictionary.com, 2011). Philosophy is another aspect of a method (Chapter 3). This chapter presents the philosophy and the basic principles of KAOS- β and EAPM. KAOS- β philosophy is strongly based on KAOS philosophy. KAOS- β philosophy focuses on the differences between KAOS- β and KAOS principles in documentation, heuristics, and aim of KAOS- β to generate traceable output for EIS architecture. EAPM philosophy is strongly motivated by processes in software architecture domain. EAPM philosophy argues over the details of activities and principles of software architecture processes and their influence on EAPM principles (structure and elements).

4.1 KAOS- β Philosophy

To develop KAOS- β an analysis of GOA is conducted and presented in Section 2.3. This analysis is the basis for the KAOS- β philosophy. Even though the philosophy of KAOS- β benefits from an empirical review of four major GOA (KAOS, GBRAM, GSN, i^*) and the lessons learnt, KAOS- β main underlying philosophy is based on KAOS. The rest of this section presents the KAOS- β philosophy that is shaped by KAOS philosophy.

To recap, KAOS heuristics were applied to an EIS example, part of stroke care rehabilitation. To facilitate following and documenting the results of applying KAOS to stroke care, the Objectiver tool was used. This helps to apply KAOS systematically and to identify its limitations, particularly in relation to modularity and traceability. Here, we highlight how KAOS has been adapted for EIS, and, in this way, outline how KAOS- β process and

philosophy differs from KAOS.

Preliminary goal identification discovered a number of top-level goals of the stroke care EIS, capturing the strategic-level aspirations of the enterprise. The Objectiver tool documents each goal by providing optional fields of Name, Def, Issues, Pattern, Category, Priority, and FormalDef. As can be seen, no source criterion is considered for the goal documentation. As in KAOS (van Lamsweerde, 2009, p. 295), the documentation of each goal differs. The common documentation includes a name (a unique identifier) and a definition, describing the goal in natural language. The definition also identifies phenomena related to the goal that could be monitored and controlled in the system. Some goal types such as *Achieve* goals consider documenting the source of a goal. In EIS goal analysis, traceability of the goals to their source is crucial, both in analysis and in the presentation of results to the enterprise. The outcome of goal analysis is a set of integrated plans (an architecture) for EIS; it is the starting point for system-level analysis and design activities, and these must be able to trace back to the sources used in devising the EIS architecture. To support traceability, the KAOS- β process adds a mandatory *source* feature to the documentation of every goal. This identifies the document pages/paragraphs that are the origin of each goal.

In KAOS, optional features of each goal are *Type*, *Category*, *Source*, *Priority*, *Stability*, *FitCriterion*, *FormalSpec*, and *Issue* (van Lamsweerde, 2009). Because the level at which KAOS- β is used (EIS architecture development rather than system-level requirements engineering), features such as *FitCriterion*, which relate to measurability of goals, are of limited value. However, contextual information is often needed: for example, it is useful to identify which specific stakeholders have an interest in a goal, and goals may include terms or assumptions that need elaborating or disambiguating. Earlier application of GSN to the stroke care example (Tabatabaie et al., 2010) had made significant use of the GSN *context* concept (Kelly, 1998b), and this has been incorporated into the KAOS- β goal documentation. KAOS- β also adds the *Issue/Notes* feature, to record goal information that does not fit in any other field, perhaps regarding future consequences or the need for further information. With these modifications, KAOS- β uses the same notations for forms and models as KAOS (van Lamsweerde, 2009). A tabular format is used for recording goal features: an example of KAOS documentation is presented in Table 4.1,

The documentation of goals leads to identifying further goals, as well as the modification or removal of some goals. Whilst iteration may be implicit in the KAOS guidelines, in KAOS- β , we make the possibility of iteration explicit, to help analysts in applying the process.

Name	MinimumInteractionWithParticipants
Def	The number of interactions between an invited participant and the system should be kept as small as possible during meeting scheduling.
Type	Soft goal
Priority	Medium
FitCriterion	At most one participant interaction about constraints in at least 80% of cases

Table 4.1: Examples of goal features model annotations (van Lamsweerde, 2009, p. 295).

In KAOS, consideration of goal categories (e.g. safety category) contributes to goal identification (van Lamsweerde, 2009). The KAOS categories (behavioural versus soft goals, and their sub-categories) refer to system-level requirements, and are less helpful at the strategic enterprise level. Instead, we use a modular structuring concept, and we apply it during the refinement of top-level goals to derive sub-goals. In the stroke care example, goal refinement led to goal-bloat – the graph of goals and sub-goals becomes unmanageable. KAOS- β modularity is inspired by GSN modularity. In the stroke care goal analysis, modules are identified via participant or stakeholder groups: health care specialists (doctors, nurses, ambulance staff etc.); community members (patients, family of patients etc.); stakeholders who deal with systems development (IT specialists, domain specialists, decision makers etc.). This leads to modules of relevance to the enterprise as well as to the EIS design activities, such as an IT module and a social module. Figure 4.1 presents an example of the modules identified for stroke care. The modules cover different aspects of stroke care EIS in its environment. The top-level modules are based on different policies that should be addressed by this EIS such as IT aspect, Social aspect, and Medical aspect. The justification for this grouping is that stroke care EIS is a governmental project and affects different groups of people in the society. Collecting the goals that address these different policies requires the engagement of different groups of experts. Because of the domain of this research, which is software engineering, the focus of the goal model is more on the IT goals. For this kind of multi view EIS, different grouping and modularity could be justified.

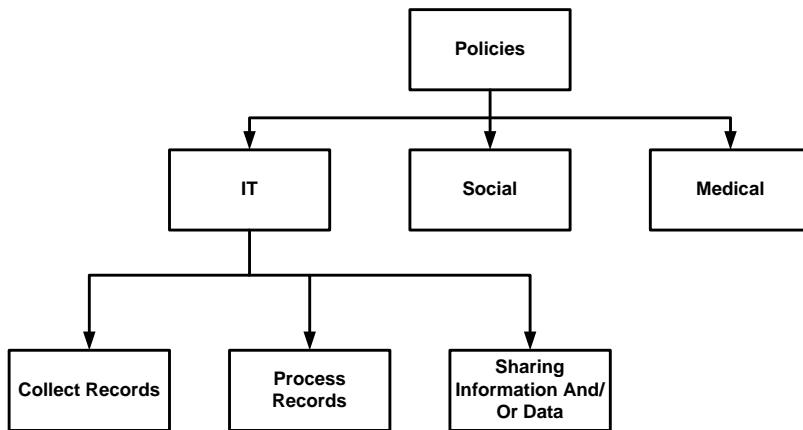


Figure 4.1: Modules derived from the stroke care EIS, from a policies' angle

Similar to identifying and structuring the goals, identifying and structuring modules is an iterative task. During the investigation process, one module could disappear or a new module could be created. Designers could change the module viewpoint after identifying more information and goals about an EIS.

A KAOS goal model records the refinement links between goals. In KAOS, a link is documented with details of the refinement that has been made: *Name*, *SysRef*, *Status* and *Tactic* (van Lamsweerde, 2009). In KAOS, the name feature is used to remove ambiguity. System reference (SysRef) refers to system-to-be or system-as-is. Status records whether the goal is still under refinement. Tactic records the refinement tactic used to derive the sub-goal (van Lamsweerde, 2009). Refinement tactics could be developed using the designers' experience. However, in KAOS- β , the *tactic* feature also records the source of a refinement (e.g. particular standards, designers' experience, or domain experts' suggestions). The refinement is motivated by explicit strategy in breaking the argument in GSN. This supports traceability, and also helps in evaluating the structure of a goal model.

In KAOS, the heuristics of goal refinement results in detection of agents, consideration of the wishes of agents, and the allocation of goals to agents. In the KAOS- β process model, these heuristic rules are consolidated into two steps, to identify EIS agents and to link them to goals. The identification of agents is closely related to the identification of personnel and stakeholder groups that form the basis of KAOS- β modules. Whilst KAOS- β uses the KAOS heuristic of asking "who" questions, to identify agents, it does not use any system modelling (KAOS proposes sequence diagrams etc. to relate

goals and agents) because this is inappropriate to the EIS level. KAOS- β uses KAOS agent categories: in the stroke care case, non-human agents are related to the operational context of the EIS, whilst the human agents are different groups of users and stakeholders: Patients; Health specialists (doctors, nurses, etc.); System developer; System maintainer; Decision makers (Hospital Managers, Government, Domain experts, etc.); Health Staff (system operators, data entry personnel, etc.).

The KAOS heuristics advise refining goals until there is one agent per goal. In enterprise goal analysis, it is important to record responsibilities, but it is unnecessary (and inappropriate) to break down goals in such detail, since these are not (yet) the basis for system-level transactions. In the enterprise design level, there may be many agents for each goal.

In KAOS, the heuristics and patterns related to *analysing obstacles, threats and conflicts* are part of goal refinement. This interesting step allows the designers to detect and address (through additional goals) interactions among goals and requirements (van Lamsweerde, 2009). In KAOS- β , the analysis of goal interaction and the resultant iteration is raised to the status of a step in the process. For example, the stroke care strategy documents yield conflicting goals relating to the ease of access to patient data (e.g. by patients and for emergency stroke treatments) and data security (e.g. the need to limit who can access personal data). As in KAOS, conflicts are identified in the goal model diagram.

The difference between KAOS and KAOS- β are summarise in Table 4.2. This summary is based on the criteria of primary goal identification, goal documentation, refinement document, agent, and evaluation checks. As can be seen in this table, to document the links and refinement the significant change between KAOS and KAOS- β is the enforcement of adding sources to create traceability. In addition, to identify agents, the important change is considering several agents to several goals. Finally, for the checks criterion, KAOS- β considers some checks that already exist in KAOS. More checks could be defined by collecting best practice and applying KAOS- β to more examples of EIS.

KAOS Goal Model	KAOS- β
Primary Goal Identification	
KAOS approach	Adds understanding of GBRAM
Implicit iteration	Explicit iteration
Goal categories (behaviour, soft goals)	No categories
No modularity	Adds modularity from GSN
Goal Documentation	
Fit criterion	No fit criterion
No context	Context (from GSN)
No notes	Notes (flexibility)
KAOS notation	Same
Refinement Documentation	
Tactics describe the refinement	Adds source in tactic (traceability)
Agent	
3 steps: detect agents + wishes + allocate agents to goals	2 steps: identify agents + allocate agents to goals
Categorise agents	Same categories
Sequence diagram to identify agents	Inappropriate in EIS level
One agent per goal	Several agents to several goals
Checks	
Converse of achieved goals	No such categories
Confusing goals and operations	To be considered
Confusing and-or refinement	To be considered
Abstract goals until reaches system boundaries (check if agent is outside the boundaries)	To be considered
Avoid ambiguity in goal specification	Not here (It is in the checklist-evaluation to make process shorter and to the point)
Stopping rule: If agent is outside the boundaries	Continue until designer is confident to start the next phase

Table 4.2: Comparing KAOS and the new KAOS- β process.

So far the KAOS- β philosophy and how it is influenced by KAOS is discussed. The next section presents the philosophy of EAPM.

4.2 EAPM Philosophy

This section presents the EAPM philosophy that is based on software architecture activities and solutions. EAPM philosophy also benefits from the lessons learnt from analysing EA solutions. However, the starting point for EAPM is the activities presented by (Rozanski & Woods, 2005, p. 24).

(Rozanski & Woods, 2005, p. 24) introduce the core concepts of software architecture as: stakeholders, architectural description and architectural elements. In Chapter 2, Figure 2.5 presents the activities supporting architecture definition. To build an architecture definition, Figure 2.6 presents the details that needs to be considered. This section presents the knowledge, that is developed from activities in Figure 2.5, and how it motivates EAPM tasks. In the following, Roz-process refers to the process and activities defined by (Rozanski & Woods, 2005, p. 24). Roz-process starts by defining the scope, behaviour and responsibilities of the system. The input is information about the organisation's needs and vision, organisational strategy, and its IT structure. Therefore to design an EIS architecture, the first step is to collect information about the goals of the organisation and additional information like boundaries, organisational strategies etc. The needs and vision of the organisation is transformed as the goals of the organisation in KAOS- β , hence the output of KAOS- β is the starting point for EAPM. The Roz-process continues with identifying stakeholders; this is aligned with the task of identifying human agents in the KAOS- β process model. The next step of Roz-process is to understand the stakeholders' concerns. This phase is transformed as the link between the agents and the goals because each goal is related to a particular agent and, based on the information gathered from the agents, the goals could be prioritised. The next step of the Roz-process is to create the architecture description. This phase is closely related to the architecture design process and it continues to the next step: to create a working architecture. More detailed SWA processes also contain an iteration between architecture design and revisiting the requirements; consider that this is the traditional SWA that relies on the information of the requirement's document.

Another process for developing an architecture is introduced by (Bass et al., 2003). The process starts with determining the architectural drivers. The process continues with prioritising the architectural drivers and applying the Attribute Driven Design (ADD). In this method, the architect breaks the

system into modules and in each module, after identifying the high priority quality attributes, some quality scenarios would be defined. Breaking an EIS into modules also is addressed by KAOS- β to present the EIS goals from different views, and to deal with the EIS complexity by focusing on smaller size and focused group of goals. Defining scenarios also has been an implicit source of knowledge for eliciting goals. Analysing this ADD activity illustrates that explicitly defining scenarios that used to capture the goals could be used for the EIS architecture to identify and support quality attributes.

ADD continues by choosing architectural patterns or strategies to satisfy the quality attributes. Architectural strategies are independent from architectural styles and provide a focused solution to address each quality attribute. This ADD activity provides the base knowledge for defining a similar task in EAPM. After identifying strategies, an architect could use architectural styles to define a connection between different elements of an architecture (i.e. goals, quality attributes, strategies) and present an EIS architecture.

By breaking modules into functional child modules that come from use cases the system breaks down into more detailed and refined modules. Each of these modules has interfaces that allow them to interact with other modules. ADD also adds constraints on child modules using scenarios; therefore, scenarios should cover both negative and positive aspects of the main functionality and quality attributes of the system. This is an iterative process for each parent module.

Analysing these processes illustrates that the common practice for designing an architecture for an EIS is to define a set of quality attributes or architectural drivers, and after understanding the attributes and drivers the designer could choose which architectural approach, *Architectural Strategy*, and *Architectural Style* could be suitable. As this is a common step for starting all these techniques, the EAPM development uses the software architecture approach with little modification to demonstrate the benefits of using goals in designing the architecture. Note that using goals in identifying quality attributes does not prevent use of other techniques such as DODAF or SOA.

To conclude, by analysing software architecture activities and practices, some of the elements for EAPM has been identified. The next two sections present these elements, their relationship and the structure of EAPM.

4.3 Conclusion

In summary, Chapter 4 presents the philosophy behind the identified elements and structure of KAOS- β and EAPM.

The KAOS- β philosophy is strongly based on KAOS philosophy, hence this chapter analyses and compares these two philosophies. The main lesson learnt during defining KAOS- β philosophy and comparing it with KAOS philosophy is that the experts of these techniques have a strong influence on how these techniques are understood and applied. Unfortunately, there are not many evidences on how these techniques have been used in practice, therefore, the main resource for making this comparison is the published documentation of KAOS. The similar argument applies for the EAPM. Architecture design in general, strongly depends on the experience of the architecture, even though there are various patterns in different domains to guide the practitioners.

Experience in using GOA is an important factor in their successful application, in addition, tools helps to generalise and extend KAOS- β and EAPM to a wider audience. Tools help to demonstrate the elements, structure, and philosophy of a process model in practice. Thus Chapter 5 presents KAOS- β and EAPM tools.

Chapter 5

Method: Tool

Another aspect of a method is a tool and language or notation to support a process and present the results of a method (Section 2.6). This chapter presents the tools developed for KAOS- β and EAPM to support these process models.

5.1 Eclipse Process Framework

Two distinct tools have been developed to support KAOS- β and EAPM. These web-based tools are implemented using the Eclipse Process Framework (EPF). EPF is an open source tool platform for process engineers and project managers to author, tailor, and publish method and processes (Haumer, 2011); it is based on the latest version of SPEM (see Section 2.4). EPF can be used without full knowledge of Eclipse, and provides a process editor that supports different breakdown structure views and graphical process presentations (Haumer, 2011): this makes it accessible to non-expert users.

The result of using EPF to produce tool support is a static website and an XML description of the process model. The benefit of using this framework is to apply the SPEM standard easier and to produce a free tool that can be sent to the users and other designers for direct use or evaluation. The EPF tool supporting KAOS- β is called Kestrel. A snapshot of Kestrel is shown in Figure 5.1 and a snapshot of the EAPM-tool is presented in Figure 5.2.

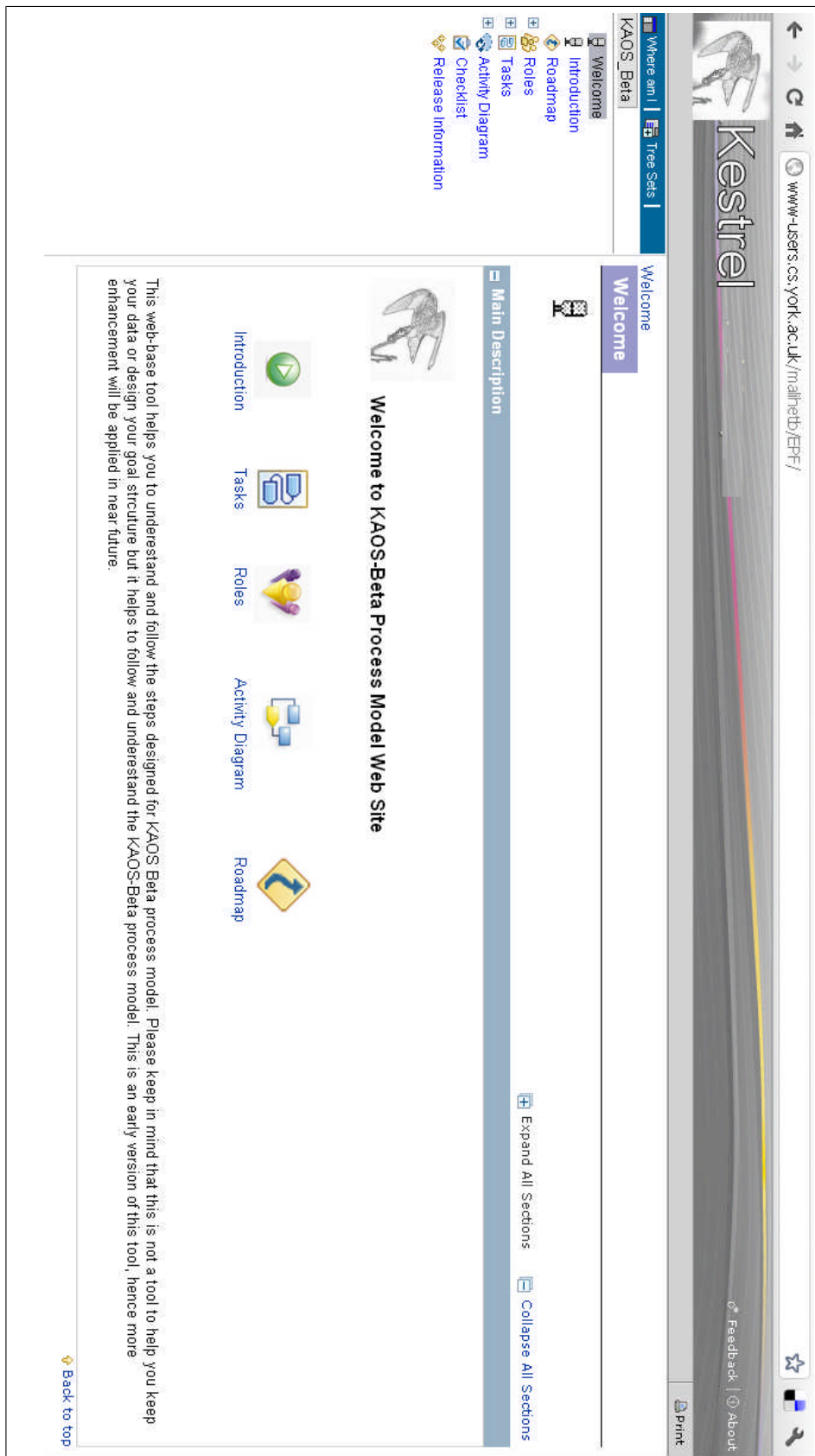
These tools present the SPEM elements related to KAOS- β and EAPM; elements such as roles, tasks, and the relationship between these elements. In addition they present the input and output of each task and the activity diagram.

In both these tools, the roles including agents in the process are identified and explicitly presented. Each role is allocated to the tasks. Tasks also are

linked to each other, this way the input and output of each task is explicit. This also helps to maintain a smooth transition between the tasks. To have a visual image of the smooth transition and links between the tasks, an activity diagram. The extra information including checklist and release information (Figure 5.1) could assist the tool users to find required information related to the process and tool (e.g. checklist of necessary activities and tasks)

Figure 5.2 presents an instance break down of the EAPM tool, which focuses on presenting a task and its related information.

In summary, the tools are one aspect of using and presenting the KAOS- β and EAPM. In addition, the KAOS notation, as presented by Objectiver (Objectiver, 2010), is used for KAOS- β . To present the information system quality attributes of EIS example using EAPM, the software architecture notation is used (Bass et al., 2003).

Figure 5.1: A snapshot of Kestrel, the tool support for KAOS- β

Eclipse Process Framework Composer

Where am I | Tree Sets | Feedback | About

new_custom_category

- Consolidate Input
- Architecture_Attribute
- Identify Measures
- Architectural Strategy
- Strategy Combination
- Architectural Style
- Candidate Architecture
- Architect
- Domain Expert
- IT/Domain Expert

Task: Architecture_Attribute

Determine the architectural drivers/attributes (less than 10)

Purpose
Collect architectural information

Relationships

Roles	Inputs	Outputs
Primary Performer: <ul style="list-style-type: none"> Architect 	Mandatory: <ul style="list-style-type: none"> QA Knowledge 	List of Quality Attributes and Drivers
Additional Performers: <ul style="list-style-type: none"> IT/Domain Expert 	Optional: <ul style="list-style-type: none"> None 	

Main Description

By analysing the collected data from previous task, some architectural information such as quality attribute and drivers is generated.

Expand All Sections | Collapse All Sections

Back to top

Figure 5.2: Snapshot of the EAPM-tool that is developed by EPF to support EAPM.

5.2 Conclusion

In conclusion, Chapter 5 briefly presents tools to support this thesis method by supporting KAOS- β and EAPM. These tools are available online and have been presented to process modelling experts. Improvements to the tools are discussed in Chapter 9.

The main advantage of using EPF is that it quickly enables development of a tool to support a process model. Another main advantage of using EPF for this thesis is to exploit SPEM standards. In practice, after developing these tools, KAOS- β and EAPM were modified to align with SPEM standards more accurately. This way the process model developers have more direct instruction on how to define a process model and not just a set of documents.

At the time of this study, EPF has not been widely used to define processes and process models; hence very few documentations helped the process authors to use EPF. However, recently EPF has received more attention in the academic domain and we produced an online step by step tutorial on how to use it (Tabatabaie, 2010); this tutorial also includes our experience of using EPF to develop KAOS- β .

We have now presented the method's processes, philosophy and tools. The next step is to illustrate the application of the method; Chapter 6 presents the results of this on the stroke care example introduced in Section 2.7.1.

Chapter 6

Method: Illustration

The objective of this chapter is to illustrate the method (particularly focusing on KAOS- β and EAPM) on a stroke care EIS. This example was introduced in Section 2.7; as mentioned earlier, the example is focused on the rehabilitation aspects of a hypothetical stroke care EIS.

The main body of this chapter contains two sections. Section 6.1 presents the results of applying KAOS- β . Even though KAOS- β has been applied fully on the stroke care example, and the full results influenced the development of Kestrel tool (Chapter 5), this section only presents a selected number of goals and their structure. The aim is to illustrate the influence of these goals on EIS architecture while using EAPM. Hence, to keep the focus on the goals, Section 6.1 does not present all the other elements of KAOS- β , such as agents and conflicts between goals. These elements are presented in Chapter 7, where KAOS- β is applied in two iterations on another example of EIS.

Section 6.2 presents the result of applying EAPM on the output of KAOS- β from Section 6.1. Section 6.2 presents the application of each task in EAPM. This complete demonstration of using EAPM leads to a hypothetical EIS architecture that is developed based on the goals of EIS and uses an EAPM process model.

Last, Section 6.3 presents a summary of the outcome of this chapter, and a number of lessons learnt from applying the method.

6.1 KAOS- β Example

The objective of Section 6.1 is to illustrate the tasks of KAOS- β , using an EIS example. The application of KAOS- β produces a set of goals. These goals are used as input to EAPM. This section presents a selection of goals

only and does not attempt to summarise all of the documentation suggested by KAOS- β .

The top goals identified for the stroke care EIS are limited to the ones identified by (Schwamm et al., 2005). Some of the goals are also supported by the scenarios defined by IBM for the Denmark electronic health system (IBM, 2005). Because of the success of the final results achieved by IBM for Denmark health system, for the rest of the goal scenarios in this section, where possible, we follow the structure used by (IBM, 2005) to define the scenarios. In these scenarios, Peter is a patient who had a stroke and now uses the stroke EIS for stroke rehabilitation phase. None of the identified goals have conflicts with the policy document presented in (DH Stroke Policy, 2007).

The stroke care EIS, in particular the rehabilitation phase, is chosen because its characteristics match with the criteria for EIS presented in Section 2.1.1. Development of a stroke care EIS is influenced by enterprise strategy and by the different views and aspects of health professional and society members. These different views shape different modules in KAOS- β .

Once goals are structured into different modules, the illustration focused on the information system applications module and its specific goals. We do not address all the information system goals, as this is impossible in such a thesis.

In task 1, according to (Schwamm et al., 2005) the top goal of a stroke care EIS is to improve stroke care prevention, treatment, and rehabilitation by providing education, undertaking research, and developing applications.

These action keywords and system intentions lead to identifying the top goals and refine it to three child goals. In task 2, each of these child goals leads to identifying a module, research module, education module, and application module.

Even though documenting the goals is the task 4 in KAOS- β process model (Figure 3.3), in practice and for a clear presentation of the progress of goal identification and refinement, we fill the forms' elements during the identification and refinement process. Therefore, Tables 6.1, 6.2, 6.3 and 6.4 present the four top goals and Figure 6.1 presents the refinement of these four top goals (task 5).

ID	SCGT1
Name	Improve Stroke Care
Def	Improve stroke prevention, treatment, and rehabilitation
Scenario	A region or country (i.e. US, UK, Denmark) uses stroke care EIS to “adequately integrate various facilities, agencies, and professionals to collaborate closely and provide stroke care services”.
Context	
Priority	High
Source	(Schwamm et al., 2005, p. 690)
Issue/Notes	Stroke is the third leading cause of death in US and UK (Schwamm et al., 2005; DH Stroke Policy, 2007)

Table 6.1: Structured documentation for goal with ID: SCGT1

ID	SCGT11
Name	Research
Def	Undertake medical, technical, and social research to improve stroke care
Scenario	Health care scientist undertake research on medicines to make patients condition stable after a stroke.
Context	
Priority	High
Source	(Schwamm et al., 2005, p. 694)
Issue/Notes	

Table 6.2: Structured documentation for goal with ID: SCGT11

ID	SCGT12
Name	Education
Def	Provide education for the stroke care system staff and society members on how to prevent, treat, and rehabilitate a stroke case.
Scenario	Peter visits the stroke EIS, “he has access to a wide range of general information from the Health Services”.
Context	
Priority	Medium
Source	(Schwamm et al., 2005, p. 694)(IBM, 2005, p. 25)
Issue/Notes	

Table 6.3: Structured documentation for goal with ID: SCGT12

ID	SCGT13
Name	Application
Def	Develop applications to support the prevention, treatment, and rehabilitation activities and services provided by a stroke system.
Scenario	Peter and his GP have access to the stroke care EIS to monitor and control his progress after a stroke. GP and other authorised health specialist use the stroke care EIS to collaborate and share data, request data, or request tests to monitor Peter's progress.
Context	
Priority	
Source	(Schwamm et al., 2005, p. 690)
Issue/Notes	

Table 6.4: Structured documentation for goal with ID: SCGT13

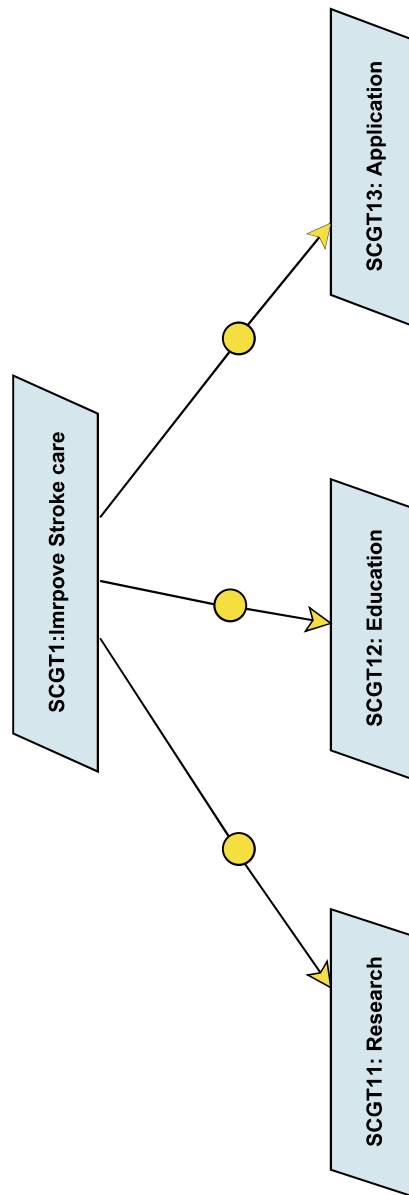


Figure 6.1: The structure of the parent goal and its three child goals.

ID	SCGL20
Name	Resource Management
Def	“A stroke care system should coordinate activities and resources to ensure that the appropriate patients are receiving appropriate care from the appropriate providers in the appropriate amount of time”.
Scenario	Peter is discharged from hospital to be under the care of a carer at home, and short term rehab devices are sent to his home.
Context	short term here is less than 10 years.
Priority	High
Source	(Schwamm et al., 2005, p. 692)
Issue/Notes	

Table 6.5: Structured documentation for goal with ID: SCGL20

For task 3, to demonstrate the KAOS- β goal documentation and trace some of the goals to the EIS architecture, we limit the focus of the goal identification to selected child goals of **Application Goal**, which earlier also is called application module. The child goals of **Application Goal**(Table 6.4) belong to part of the stroke rehabilitation services and activities. Note that some of the goals identified for rehabilitations phase could be valid for stroke presentation or stroke treatment, but at this stage, to limit the example, the scenarios are limited to stroke rehabilitation.

In task 4, the identified goals are documented in Table 6.6 to 6.13; they are used as an input for the next process model, EAPM, in Section 6.2.

Figure 6.2 also presents task 5, which is the goal’s refinement. As mentioned earlier in this chapter, the results of task 6 onwards are not presented in this section. This is because these tasks are demonstrated in another example of EIS (Chapter 7), and because, to identify agents and conflicts between goals, further goal identification is required. However, the objective of this section is to identify enough goals that could be used as an input for EAPM. The next section presents the influence of the current identified goals on the EIS architecture.

ID	SCGL201
Name	Clear Protocol
Def	Define clear transport protocols for providers to ensure that patients are taken only to facilities with appropriate and sufficient resources.
Scenario	Peter's carer notices the symptoms of another stroke, he calls the emergency number, the ambulance system has the record and address of Peter and pick him up from his current location (assuming he is at his current location) and deliver him to the suitable emergency room.
Context	
Priority	High
Source	(Schwamm et al., 2005, p. 692)
Issue/Notes	This is just one of the many possible child goals for Resource Management parent goal that is focused on transport protocol.

Table 6.6: Structured documentation for goal with ID: SCGL201

ID	SCGL21
Name	Communication
Def	Enhance communication among rehabilitation team, patient, patient's carer, hospitals and emergency medical services (EMS).
Scenario	The laboratory updates the test results of Peter, thus he and his carer receives a notice for the changes in his records. Peter also is informed about the new test that he need to do and he should contact his local rehabilitation team to discuss his status.
Context	
Priority	High
Source	(Schwamm et al., 2005, p. 692)
Issue/Notes	

Table 6.7: Structured documentation for goal with ID: SCGL21

ID	SCGL211
Name	Communication Patient & Health Specialist
Def	Provide communication services between patients and other healthcare professionals.
Scenario	“Peter decides to give his local pharmacy consent to view his personal medicine profile, so they may assist him better when advising him in the use of medication”.
Context	
Priority	High
Source	(IBM, 2005, p. 38)
Issue/Notes	

Table 6.8: Structured documentation for goal with ID: SCGL211

ID	SCGL212
Name	Communication Systems
Def	Provide communication between stroke related health system applications (i.e. patient electronic record, laboratory system, scanning system, ambulance system, telemedicine).
Scenario	Based on the results of his latest lab test, Peter records are updated and he receives a number of suggestions for his food diet.
Context	
Priority	High
Source	(Schwamm et al., 2005, p. 692)
Issue/Notes	

Table 6.9: Structured documentation for goal with ID: SCGL212

ID	SCGL213
Name	Communication Rural Area
Def	Provide stroke care services for patients who live in rural and remote areas.
Scenario	“Since Peter (a patient) can access the stroke care EIS portal anywhere in the world, he is also free to travel, and the clinic can monitor his condition remotely and only needs to see him once a year”.
Context	
Priority	High
Source	(Schwamm et al., 2005, p. 691)
Issue/Notes	

Table 6.10: Structured documentation for goal with ID: SCGL213

ID	SCGL22
Name	Security
Def	Provide a secure stroke care EIS for the use of stroke system members.
Scenario	Peter and health specialists connect to stroke care EIS using different devices to communicate and send the changes about his stroke progress. They also make decisions based on the information provided by the stroke care EIS.
Context	
Priority	High
Source	(IBM, 2005, p. 27)
Issue/Notes	

Table 6.11: Structured documentation for goal with ID: SCGL22

ID	SCGL221
Name	Data Security
Def	Provide the most appropriate data for stroke EIS functions.
Scenario	peter has an appointment with his doctor. Before the meeting “Peter’s doctor retrieves Peter’s EPR from the general database, and is updated with information from Peter’s hospitalisation and anti-coagulant therapy”.
Context	Appropriate data here means up to date and not corrupted data.
Priority	High
Source	(IBM, 2005, p. 22)
Issue/Notes	

Table 6.12: Structured documentation for goal with ID: SCGL221

ID	SCGL222
Name	Access Security
Def	Provide easy and secure access to personal healthcare information for the authorised members.
Scenario	“Using his digital signature, Peter logs on to his personal page on the portal. Here, he finds information related to his own interaction with the health services”.
Context	
Priority	High
Source	(IBM, 2005, p. 28)
Issue/Notes	

Table 6.13: Structured documentation for goal with ID: SCGL222

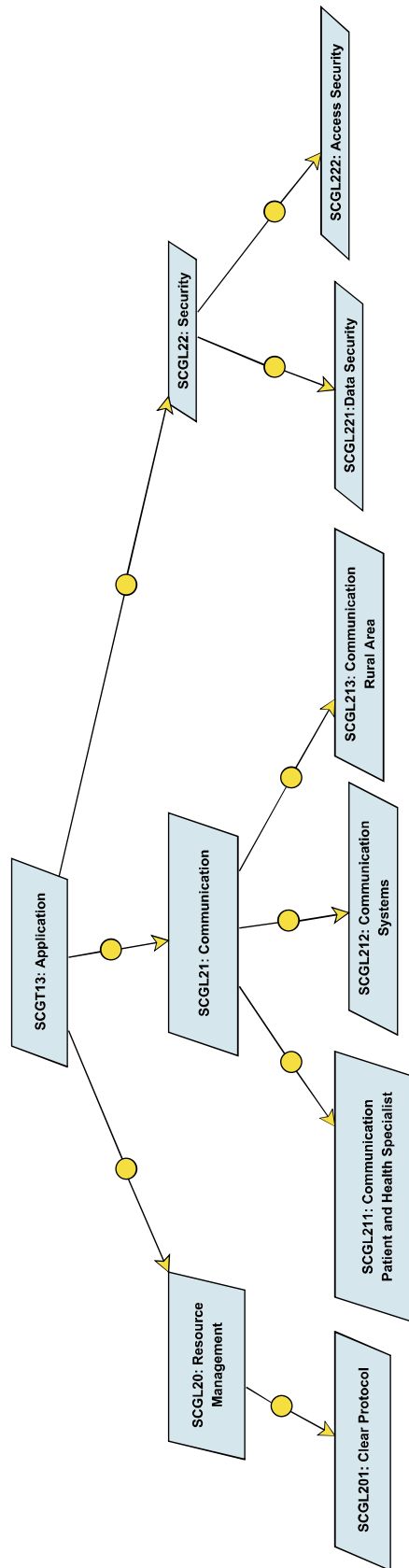


Figure 6.2: The structure of application goal refinement.

Quality Attributes	Goals
System Responsiveness	SCGL201,21
Availability	SCGL213,221
Security	SCGL22,221,222
Freshness	SCGL221
Ubiquitousness	SCGL222
Usability	SCGL213

Table 6.14: Relationship between goals and quality attributes. Numbers are referred to the goals in Section 6.1.

6.2 EAPM Example

This section presents the EAPM tasks applied to the stroke care EIS example. The main input for EAPM for this example are the goals identified and structured using KAOS- β in the previous section.

6.2.1 EAPM: Tasks One and Two

The aim of these steps is to consolidate the inputs of EAPM, including the goals of EIS, and to determine quality attributes and drivers. The goals have already been extracted and structured using KAOS- β (see Section 6.1). Therefore using the information provided by KAOS- β , an explanation for quality attributes and drivers is undertaken.

Table 6.14 presents the quality attributes that are identified from the goals – KAOS- β 's output. The numbers under the Goals category in Table 6.14 presents the number or ID of the goals that were defined while using KAOS- β . The descriptions of these goals are presented in Section 6.1. The goal knowledge is one of the sources (here the main source) of identifying quality attributes. For example, SCGL212, which is a goal identified by KAOS- β , is identified as: enhance communication among stroke related health system applications. SCGL213, also is identified as: provide stroke care services for patients who live in rural and remote areas. To address these goals, the responsiveness quality attribute is identified. Responsiveness for this system is defined as: the system provides feedback to the users' requests regardless of their geographical position within the domain of EIS.

Following this, further quality attributes that are identified from the goals in Table 6.14 are as follows:

- **Responsiveness:** Responsiveness or data and function are availability

means system gives feedback to the users' requests regardless of their physical position.

- **Availability:** Service interruptions in EIS functionality minimised or removed. This quality attribute and responsiveness derive from the same goals. A description of a goal could lead to more than one quality attributes.
- **Security:** EIS data and functionalities should be available in a secure environment.
- **Freshness:** The most appropriate data for each EIS function exist in the EIS. Data freshness and data accuracy could be categorised under security quality attribute too (Perrig et al., 2001; Peralta, 2006).
- **Ubiquitousness:** The EIS can be accessed by the authorised users, regardless of access device or location.
- **Usability:** After training, users of the system can use the system with no destructive frustration. Further usability studies and investigations should be provided to collect users' feedback. Usability in this study has different aspects such as user friendliness, saving time, and feel as using a single system.

So far a number of quality attributes have been extracted from the information provided by the KAOS- β goals. The next step uses the information about the system and expertise of the architect to add more architectural information such as measures of success.

6.2.2 EAPM: Task Three

The aim of the third step is to identify the measures of success for the quality attributes and, consequently, for the EIS architecture. These measures could be defined from designers' expertise, system characteristics, or standards in each field; for example security standards for health system such as ISO for confidentiality (Public Health Data Standards Consortium, 2011).

For the stroke care example, we made assumptions for the measures of identified quality attributes to progress this task. These assumed measures of success include time (e.g. less than 10 minutes), high, medium, or low to emphasise the importance of the quality attribute in a trade off analysis process. For example for availability quality attribute, availability of the system should be as high as possible (e.g. system works for users 99.999% of time (Clements et al., 2002, p. 51)).

- Responsiveness: Less than 10 minutes, but this will depend on the situation: e.g. if it is an emergency or non-emergency function of the stroke care EIS
- Availability: High (system works for users 99.999% of time (Clements et al., 2002, p. 51))
- Security: High (patients' database authorisation works 99.999% of time (Clements et al., 2002, p. 51))
- Freshness: High (data updated within the system in less than 3 minutes after changes of data)
- Ubiquitousness: At least one interface and device to use stroke care EIS inside hospital and one in ambulance should be active. At least the patients records should be available, if other functionalities are not available.
- Usability: In a normal functionality of the EIS, at least 2 different solutions for communications should be supported, this way a larger group of people have a chance to be able to use the interface with less problem (i.e. voice based interface and touched based interface)

As can be seen, the measure of success could be defined by a simple comparative measure such as high, medium and low. It also could be a statistical or quantitative measure. The aim is to make the importance and in cases resources' demands of these quality attributes more clear for the team of designers. Measures of success could help designers to choose the strategies and styles and design the architecture to address the important quality attributes. For examples, strategies may be chosen that provide high security, or provide different usability solutions. These may be conflicts between strategies (i.e. security and usability); based on the measures of success, it is important not to compromise the security. However, the system should be usable for the users. If certain devices put security at risk, they should not be considered in the strategy. These decisions could be made using the information collected from measures of success. In addition, measures of success can help designers to evaluate the quality architecture in the later stages. If a change is required to be applied to an architecture, it should be aligned with the criteria developed by measures of success (i.e. high security).

6.2.3 EAPM: Task Four

The aim of the step four in EAPM is to identify one or more strategies for modelling the architecture. There is no specific list for software or EIS architecture strategies. Some of these strategies are presented in (Bass et al., 2003) as examples. To demonstrate this task, arbitrary strategies are allocated to address quality attributes. These strategies are relevant to each quality attribute but not necessarily the best. For each quality attribute at least two different strategies are allocated to emphasise the possibility of having different architectures by choosing different combinations of strategies.

1. Responsiveness

- Local caching: improve the data availability by data caching on the client-side (Ma, Vazhkudai, & Zhang, 2009).
- Distributed replicated data: store multiple copies of data at various sites in a network and provide data availability even when the sites or communication links fail (Lazoff & Stephens, 1996).

2. Availability

- Fault detection (Heartbeat: one component emits a heartbeat message periodically and another component listens (Bass et al., 2003, p. 102).)
- Fault recovery (Active redundancy: all redundant components provide the same functionality, are always active, and respond to events in parallel; downtime is on the order of milliseconds (Bass et al., 2003, p. 103).)

3. Security

- Authenticate users (Bass et al., 2003, p. 119)
- Authorise users (Bass et al., 2003, p. 119)
- Local security responsibility

4. Freshness

- Weak freshness: To have a recent data and ensure that no adversary replayed old messages, the strategy is to have a partial message ordering that carries no delay information (Perrig et al., 2001).

- Strong freshness: To have a recent data and ensure that no adversary replayed old messages, the strategy is to provide a total order on a request-response pair, and allows for delay estimation (Perrig et al., 2001).

5. Ubiquitousness

- Mobile device caching
- Support multiple redundant communication devices (landlines, mobile phone, laptop, desktop)

6. Usability

- Separate user interface design and implementation from the rest of the application to localise expected changes during development and after deployment (Bass et al., 2003, p. 123)
- Support user initiative (cancel, undo, aggregate)

To propose an architectural solution, an architect could propose different strategies. Different combinations of these strategies creates different architectures. Different architectures should be compared with each other as to choose the most appropriate.

6.2.4 EAPM: Tasks Five, Six, and Seven

The last three steps of EAPM focus on selecting strategies, identifying styles, and producing an architecture. To accomplish these steps we introduce and design a template for presenting the architectural information.

Table 6.15 presents a template motivated by Architecture Tradeoff Analysis Method (ATAM) (Clements et al., 2002). Some of the fields are changed to fit our objective – illustrating the relationship between goals and quality attributes. The fields that are similar to the ATAM template for analysis of the architectural approach are: scenario, quality attribute, environment, stimulus, response, reasoning, architectural diagram.

The field quality attribute is the key in this template, it provides the name, if required short description, of the architectural quality attribute. Each quality attribute is supported by a scenario. Scenario also makes a quality attribute understandable for the non-IT experts. The scenarios are motivated by the goal description. Therefore, to explicitly present the source of each scenario, the field of goal is added to the EAPM template. Goal (s) and scenarios fields present the link for transition from goals to EIS architecture.

The environment field presents the relevant information about the system environment and the environment which scenario is carried out (Clements et al., 2002). The information about the environment has a valuable effect when the EIS is dealing with the change. Change is inevitable for a business-oriented system such as EIS, hence the architecture design should presents the solutions to address the changes. Table 6.15 presents a normal operation of the EIS for the environment that could be applicable for the first iteration of the design. The later iterations include special cases such as dealing with the expected changes.

The stimulus field presents a “precise statement of the quality attribute stimulus (i.e. failure) embodied by the scenario” (Clements et al., 2002). Even though this information is not requested directly from the EAPM tasks, stimulus field extracts additional and detailed information about the scenario that could clarify why a quality attribute is required.

To explain how a quality attribute responds to a stimulus, the EAPM and ATAM template have a response field. This field presents a “precise statement of the quality attribute response” (Clements et al., 2002). This field presents detailed information about effects of the existence of a quality attribute by presenting its response to the environment’s stimulus.

Reasoning is a field that is borrowed from ATAM. This field presents the qualitative and quantitative rational for how a quality attribute addresses a goal. In addition, this field could present how a strategy addresses the quality attribute. The information in this field helps the reviewers to understand or criticise why a quality attribute is chosen.

EAPM template presents two fields of measure and strategy to document the information collected from EAPM tasks three, four, and five. One source of identifying the measure is the goal structure and documentation.

To present a diagrammatic view of the chosen architectural strategies and styles that address each quality attribute and goal, EAPM and ATAM provide a field for architectural diagram.

The ATAM template also presents information about criteria such as tradeoff and risk which could be added to the EAPM if required. However, at this stage, the tradeoff and risk of particular strategies do not add any value as the strategies are selected arbitrary to demonstrate the EAPM tasks.

In the architecture description documents, the architect fills in a template form to describe each relationship between quality attributes and goals. In this chapter, the information for one quality attribute is presented in Table 6.16 as an example.

The completed template can be used by the architect to start designing the detailed EIS architecture. In practice, experts analyse different combina-

tions of strategies to choose a suitable architecture, but here the objective is to demonstrate the process of going from goals to early phases of architecture, not to design the best possible architecture.

Attribute (s)	Name of the quality attribute(s)
Scenario (s)	Sample scenario to explain the goals and quality attribute
Goal (s)	One or more goal id (from KAOS- β output)
Environment	e.g. normal operation of the system
Stimulus	A trigger event such as hardware fault or threat
Response	A precise statement of the quality attributes such as response time
Reasoning	Qualitative and/or qualitative rational of how a quality attribute addresses the goal, plus how a strategy addresses the quality attribute
Measure	Measure of success for the quality attribute(s)
Strategy	One or more strategies to address quality attribute
Architectural diagram	Cross reference to architectural views and reasoning

Table 6.15: The structure of a template motivated by ATAM template; for recording analysis of an architectural approach (based on Clements et al. (2002, p.122))

Attribute (s)	Responsiveness
Scenario (s)	A patient visits a local emergency room and a doctor inserts the patient's information and find the medical record.
Goal (s)	SCGL21, SCGL201
Environment	Normal operation
Stimulus	Insert the patient name and date of birth to the system
Response	In less than 10 minutes
Reasoning	System should be able to present at least the basic patient's medical record
Measure	Present the fresh patient's record in less than 10 minutes
Strategy	Local Caching
Architectural diagram	

Table 6.16: A filled example of EAPM results template (Table 6.15). The source of the content is output of KAOS- β presented in Section6.1.

Based on the collected information about the architecture so far, one possible model of a stroke care EIS architecture is outlined in Figure 6.3. The diagram in Figure 6.3 presents the business process layer, quality attributes, and the service layer of the architecture. The top layer and the bottom layer of Figure 6.3 are presented in detail in Figure 6.4 and 6.5.

Figure 6.3 presents the relationship between an example of a business process for stroke care EIS, related quality attributes, and further architectural information such as strategies and styles. The first layer in this figure presents a data flow diagram to illustrate a business process. In this business process, users log in and use the system to access the patients record and medical queries. This diagram is created using a design decomposition matrix (DDM) (Bassry and Associates, 2010). DDM is not an element of EAPM; it is used at this stage as a tool to model a business process. Examples of other approaches to model a business process are flowchart (Wikipedia, 2011f) and business process model and notation (BPMN) (Wikipedia, 2011c).

The business process layer is presented in more detail in Figure 6.4. This business process is based on a scenario of using the stroke care EIS. This scenario describes when authorised healthcare users (i.e. doctors and nurses) wish to use the system, users should log into the system by entering login information and if their aim is to have access to the patient's record based on their access level, they request the patient's record by entering patient's searchable information (such as name and date of birth). The further steps present possible queries of patients' record, either just part of patients' record or requesting relevant medical advice (tasks 3 and 4 in Figure 6.4).

The middle layer in Figure 6.3 presents the quality attributes that are established from early EAPM steps. The bottom layer in Figure 6.3 presents the mapping from quality attributes to the candidate strategies that are nominated to address the quality attributes. Figure 6.5 presents the name of these strategies. Figures 6.3, 6.4, and 6.5 partially present the results of EAPM's steps for an assumed scenario in stroke care system.

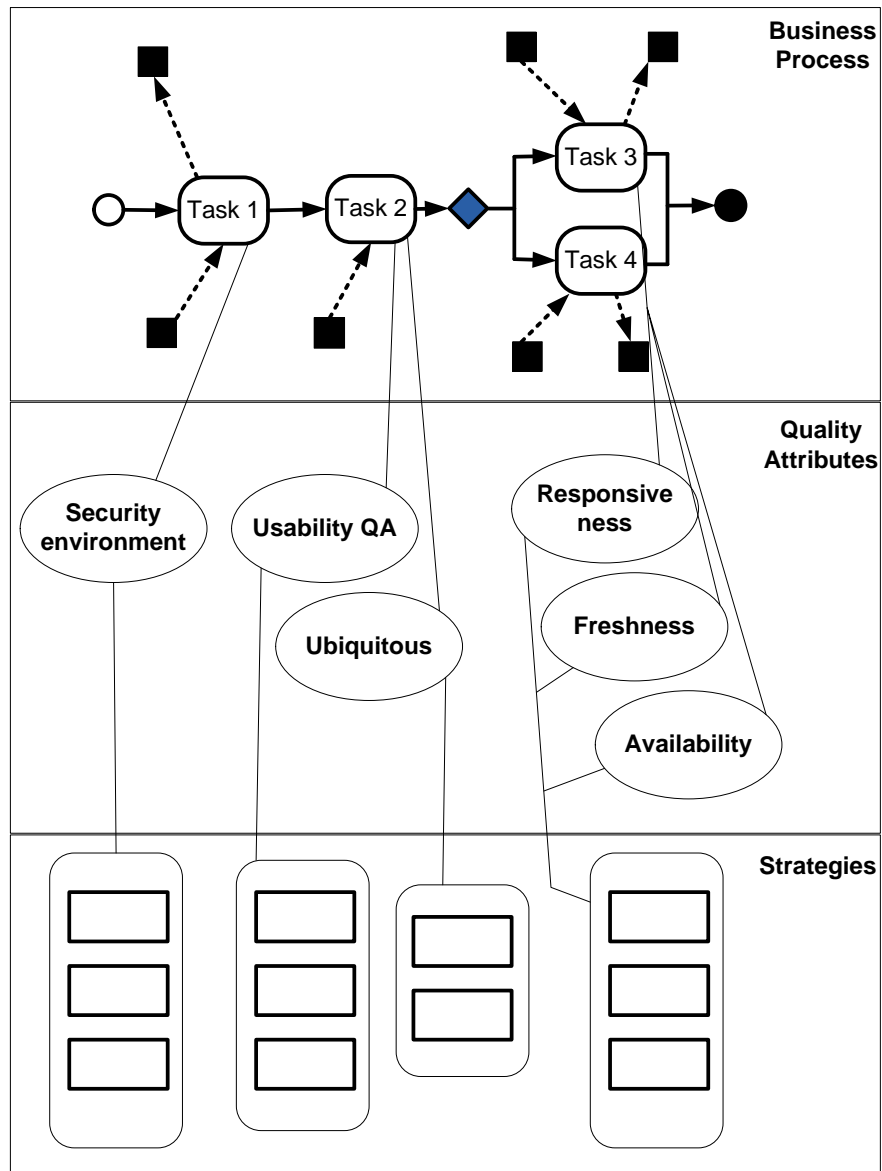


Figure 6.3: A possible model for EIS architecture presenting the links between business processes, identified quality attributes, and sample strategies. The top layer is presented separately with more detailed information in Figure 6.4. The boxes in the strategies section are presented in detail in Figure 6.5.

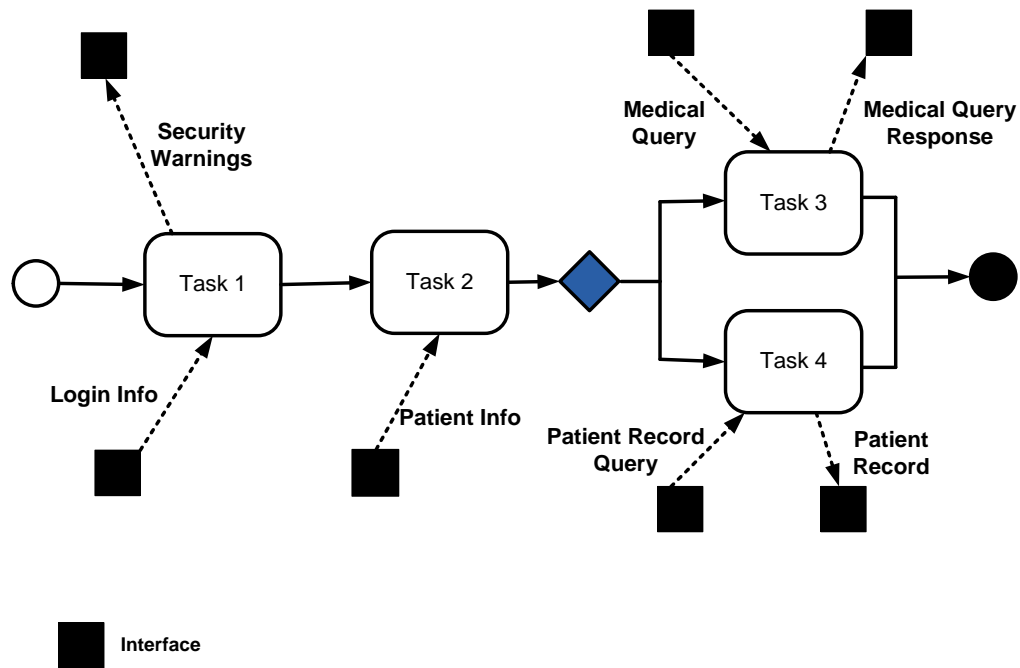


Figure 6.4: Details of a business process for a scenario of stroke care EIS. This business process is the top layer in Figure 6.3.

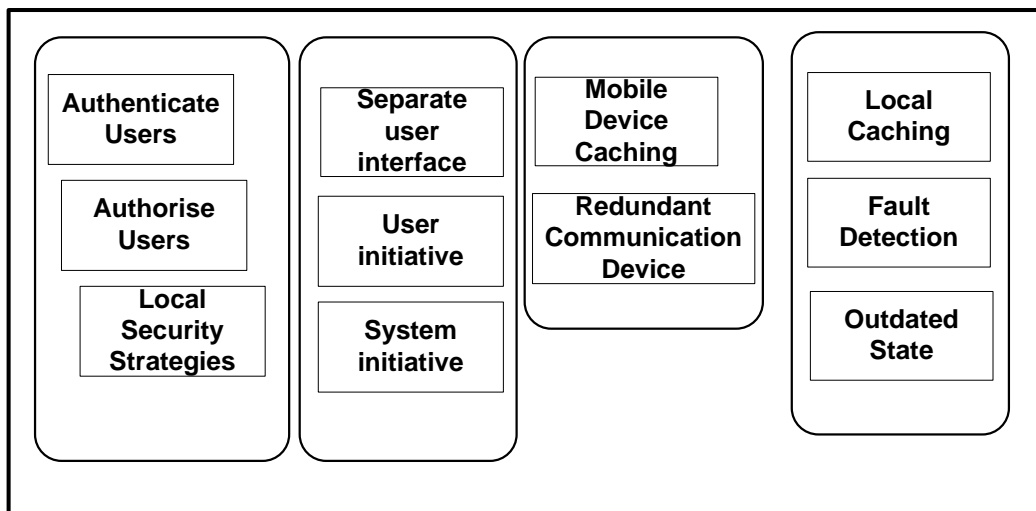


Figure 6.5: Selected strategies from Section 6.2.3. This is more information for the third layer in Figure 6.3.

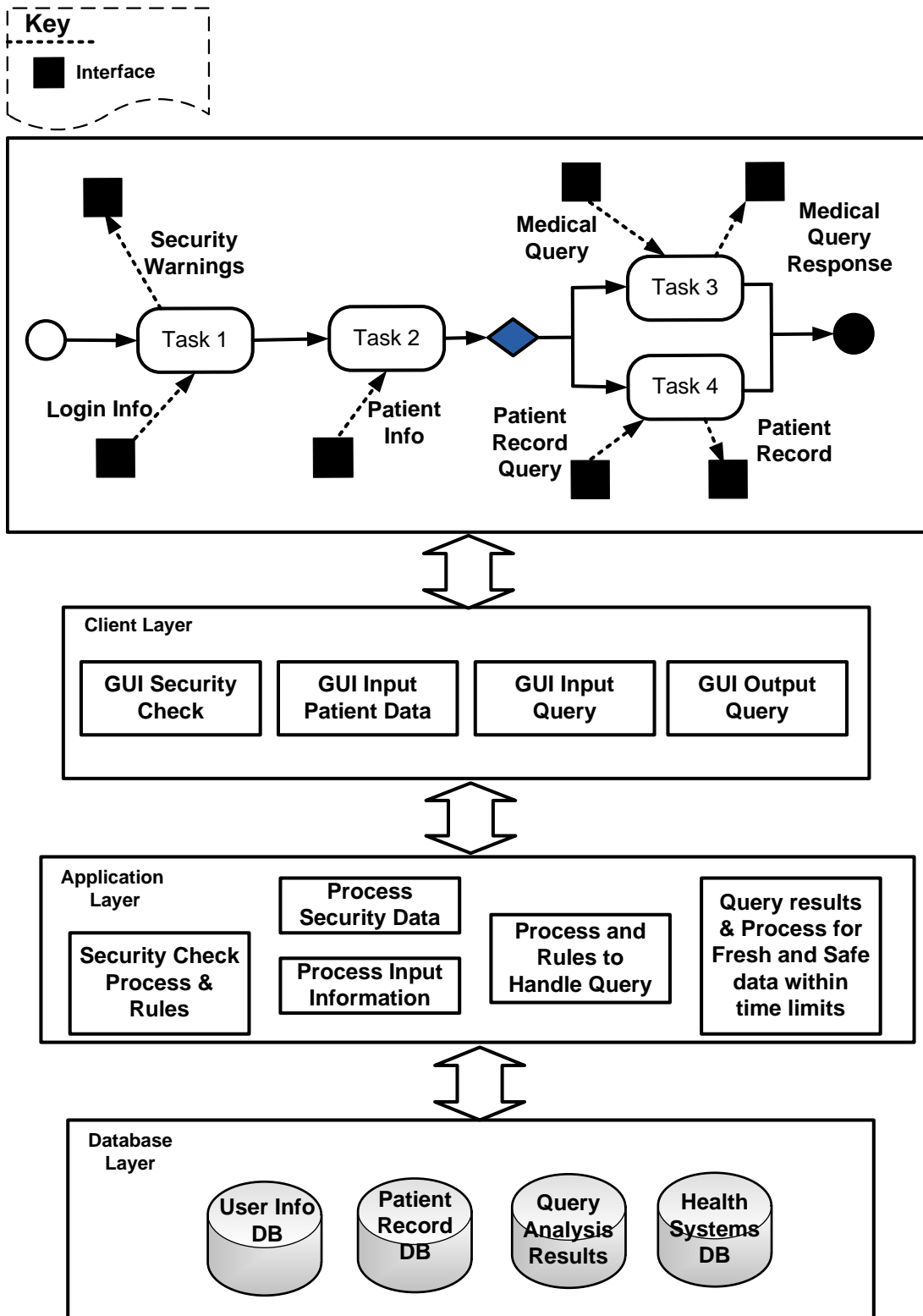


Figure 6.6: Partial architecture for stroke care EIS using three-tiered architecture style (IBM Software Information Center, 2007).

Figure 6.6 presents an alternative, more modest architecture for stroke care EIS. In this architecture a three-tiered architectural style (Rozanski & Woods, 2005) is chosen to address the previously defined quality attributes. The three-tiered style is a common design of the client/server system (IBM Software Information Center, 2007), and includes the presentation or client layer, business logic or application layer, and data layer; and have been applied to example of EIS architecture (Pradhan, Laefer, & Rasdorf, 2007). For the stroke care EIS, the interface layer includes the graphical user interface (GUI) for the applications that address the quality attributes. For example, the GUI for the security check that is mapped to the applications related to security quality attribute. The business logic layer includes different applications for addressing quality attributes, and the database layer presents the stroke care EIS database and data warehouse.

6.3 Conclusion

After defining a method in Chapters 3, 4, and 5, Chapter 6 illustrates the application of this method on a stroke care EIS example. This example was introduced in Section 2.7.1. Chapter 6 contains two main sections, illustrating the core of KAOS- β and EAPM.

In summary, Chapter 6 identifies, structures, and traces part of the top goals of a rehabilitation phase of a hypothetical stroke care EIS to a top-level EIS architecture. During the process of applying the method to the EIS example, we learned that the goal identification and documentation improves by the experience of the domain experts. This chapter demonstrates the importance of using scenarios to identify the goals. These scenarios are defined to identify and structure a number of goals that influence an EIS architecture by identifying a number of quality attributes. The quality attributes are limited to the software engineering solutions. Practitioners from different background such as health specialists, sociologist could identify further scenarios and goals that are aligned with the top-level stroke care system goals, yet address their viewpoint.

The Stroke care example is the first of two that were introduced in Section 2.7. Applying the method to more examples of EIS from different domains will increase the level of maturity of the method. Thus Chapter 7 demonstrates the second case study of this thesis: Airport Crisis Management (ACM).

Chapter 7

The Airport Crisis Management Example

This chapter presents the results of using the method on an Airport Crisis Management (ACM). The example application of the method produces a number of goals for ACM and presents the transition from goals to a possible top-level EIS architecture. The results of this chapter lead to empirical justification of the elements, structure, and philosophy of the method.

Prior to presenting the results, Section 7.1 reviews the ACM example. Section 7.2 presents the results and justification of applying KAOS- β to ACM documents in two iterations. Section 7.3 presents part of the steps, elements, and structure of EAPM on ACM example. Given what was presented in Chapter 6, Sections 7.2 and 7.3 do not repeat justification of the elements and structure of KAOS- β and EAPM. Finally Section 7.4 presents a reflective conclusion of the results of Chapter 7.

7.1 Airport Crisis Management: Background

An ACM is a collection of software and hardware solutions to give direct support to all crisis management activities: from incident reporting to logistics workflow support. An ACM system that deals with alignment of departments and business processes can be considered to be an EIS.

Like the stroke care EIS, ACM involves interactions of many departments, has various business processes that could change over the time and has impact on wider society – as in the 2010 volcanic ash cloud disruption. Its similarities to the stroke care EIS makes it appropriate for assessing the utility of the method.

Examples of co-operating departments in a crisis are police, media, hospi-

tals. The volcanic ash cloud disruption in 2010 is an example of a large-scale crisis that ACM may address. A more restricted and manageable crisis is as follows:

An airplane catches fire while refuelling at a gate, an alarm event is generated to inform the different departments. CCTV sends usual information to relevant departments, which broadcast the latest information.

As in the stroke care EIS, there are limitations in the amount of information available for applying the method. The main sources for this real example are the notes from private communication with the domain expert¹ and (MODELPLEX Consortium, 2007) which is based on a real EIS. To expand understanding of this example, we also reviewed other materials including online news articles such as (ACM, 2010; Noland, 2011). This version of the case study is designed to fit within the time limits and scope of this thesis. When there is not enough information, realistic assumptions have been made.

7.2 Applying KAOS- β

To apply KAOS- β systematically, the Kestrel tool, developed as part of the method (Chapter 5) is used. In two iterations, the steps of KAOS- β have been applied. The result of the first iteration is presented in plain text; the result of the second iteration is presented using graphical diagrams to demonstrate different techniques. In a real analysis using KAOS- β , the final documentation would present the results in both textual and graphical modes.

7.2.1 First Iteration

The first iteration applies the nine steps of KAOS- β . This is a summary of the steps and the results. The details of the steps are presented in Chapter 3.

Step 1: Identify top goals

¹R.Paige. Case Study: Airport Crisis Management. Unpublished notes.

To identify the top goals, the first step is to review the documents describing ACM system¹, and online news (ACM, 2010; Noland, 2011). The review identified the following goals (in no particular order):

1. Clear communication
2. Reliable information to all actors
3. Effective dissemination
4. Software and hardware solutions to support activities
5. Identify and register victims

Step 2: Identify modules

To identify the modules we analysed the documents about system description and scenarios (i.e. (ACM, 2010; Noland, 2011)). For example, *identifying and registering victims* is a goal that indicates a *social* module. In a large-scale crisis such as volcanic ash disruption in 2010, the media and many society members are affected. The *social* module addresses the goals, actors and activities that apply to society, media, families of victims, politicians etc. A *business* module indicates the goals, actors, and activities that support the business aspect, e.g. the financial issues and outcome of the crisis, or the lessons learnt to minimise the impact of the crisis on the life of the airport.

We identified goals that address **clear communication, software and hardware solutions to support activities**. These goals are the indicators for *IT* module. **Clear communication** goal in addition to goals addressing **reliable information to all actors, effective dissemination, and identify and register victims** are indicators for different aspects of *airport* module. The detailed analysis of the data and possible scenarios illustrates the option of sub-categorising the airport module to the four *Tower, Ground, In Flight*, and *IT* modules. *IT* module is a technical module that could help to link the other three modules together. We assume that currently there are some IT systems dealing with part of the functionalities inside this enterprise. However, no IT system that performs as a platform for collaboration of the current IT and non-IT systems and integrate IT systems with business goals, ACM EIS. As this research focuses on EIS systems, we select the IT modules for further investigation.

Figure 7.1 shows the modules identified in step 2. Note that the IT module should support the functionalities of each module as required.

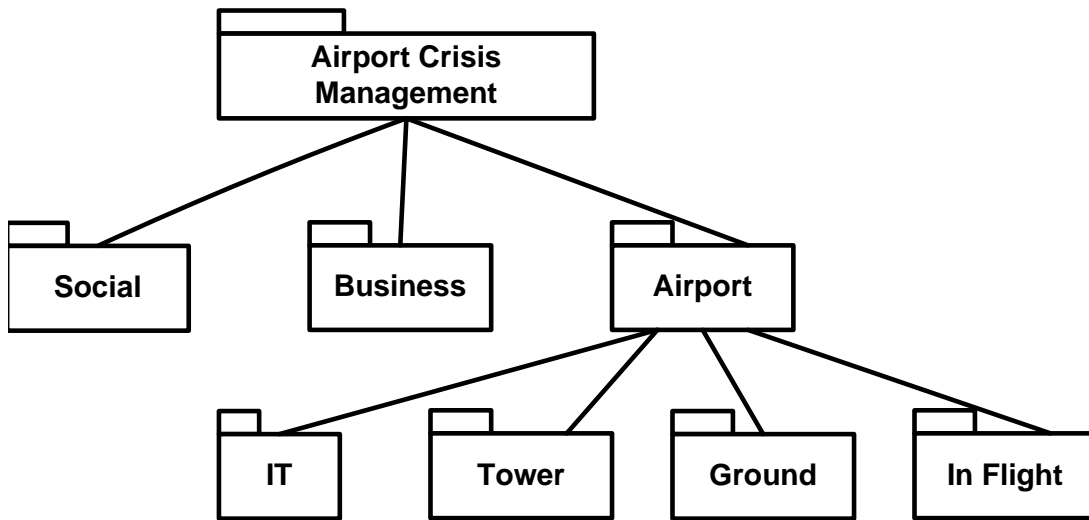


Figure 7.1: The modules identified for ACM.

Step 3: Identify goals in each module

In this step the analysis of the ACM documents focuses on goal identification within each module. The following lists elaborate the high-level goals that relate to each module:

IT Module:

- Software and hardware solutions to support activities
- Clear communication
- Identifying and registering victims
- Effective dissemination

Social Module:

- Reliable information for actors and stakeholders via public relation representatives
- Informing the families of the victims
- Sending relevant information to the media to inform society²

Business Module:

²For example in the case of volcanic ash disruption (Jordans & Lekic, 2010) the airports should inform people intending to travel and their related parties about the flight situation; this could be done via broadcast media.

- Reliable information for actors
- Allocate budget
- Prepare staff, training, facilities to deal with crisis
- Future planing based on the learning outcome of each incident

Ground Module:

- Clear communication
- Reliable information to actors
- Effective dissemination
- Identify and register victims

In flight Module:

- Clear communication
- Reliable information for actors
- Effective dissemination
- Identify and register victims

Tower Module:

- Clear communication
- Reliable information for actors
- Effective dissemination

To complete the application of KAOS- β , the rest of this section focuses on one module. However, the application considers the airport module and its sub-modules as one module, and applies the KAOS- β process to the whole airport module. The reasons for this are: firstly, the four sub-modules and the four following goals are very similar, hence they could be considered as goals of one parent module. Secondly, time and access to documents restricts the detailed information available to apply KAOS- β for the sub-modules individually. These factors affect the size of the case study but not its behaviour; even though a realistic size for a case study is desirable, it is not a necessity. Thus we continue with considering airport modules and its sub-modules as one module. Figure 7.2 presents the modules used in the KAOS- β pilot.

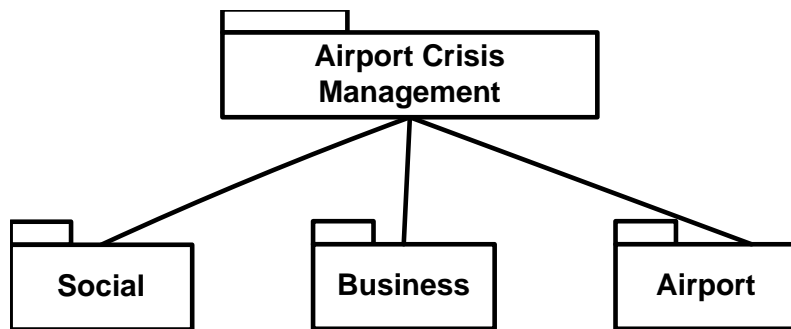


Figure 7.2: Airport Crisis Management modules for steps four to nine of the KAOS- β pilot.

Step 4: Document goals

To document the goals, KAOS- β proposes a number of forms, introduced in Chapter 3. The aim is to describe the goals clearly, justifiably, and traceable to their sources. Three out of nine primary top goals are presented in this section, to demonstrate the suggested information and structure for documenting the goals. Forms helps to structure the required information and can enhance the process of implementing systematic tools in future works that can support KAOS- β results. Table 7.1 to 7.4 are examples of how the sample data are used to fill the forms for the identified goals. The rest of the documentation for the detected goals is presented in Appendix A.1. This documentation in the later steps helps reader to understand the links between goals.

ID	ACMG1
Name	Share awareness, reliable information, full and seamless communication links
Def	A crisis management system for airports must allow shared situational awareness from the field through to command centres; reliable information from a number of sources; and shaping full, seamless communication links.
Scenario	An airplane got on fire while refuelling at a gate, an alarm event generated to inform the incident to different departments plus the CCTV sends the information to relevant departments to broadcast the latest information.
Context	Seamless means “simple and comfortable communication across various technologies” (German Telekom, 2011)
Priority	High
Source	ACM description and sample scenarios ¹
Issue/Notes	None detected at this stage

Table 7.1: Structured documentation for ACM’s goal with ID:ACMG1

ID	ACMG10
Name	Reliable Communication
Def	ACM should collect information from reliable sources and guaranteed to reach their destination complete and uncorrupted and in the order they were sent.
Scenario	After an alarm has been triggered, critical information (smoke location estimate and composition) needs to be collected from machines and human and sent urgently to the appropriate personnels (e.g., the fire department) for analysis and decision on how to limit the potential impact.
Context	“Data corruption refers to errors in computer data that occur during transmission, retrieval, or processing, introducing unintended changes to the original data” (quick guide to key technical terms, 2011)
Priority	High
Source	ACM description and sample scenarios ¹
Issue/Notes	None detected at this stage

Table 7.2: Structured documentation for ACM’s goal with ID:ACMG10

ID	ACMG2
Name	Clear Communication
Def	Clear communication procedure, standard and tools to allow departments and rescue units communicate with each other under defined security levels
Scenario	The crisis management system support secure, fast and reliable communication activities and provide information about the stakeholders that can assist with the situation.
Context	
Priority	High
Source	ACM description and sample scenarios ¹
Issue/Notes	Measure of success: security, reliability, and providing stakeholder information within specific period of time.

Table 7.3: Structured documentation for ACM’s goal with ID:ACMG2

ID	ACMG3
Name	Cross regional involvement of Crisis Management Departments
Def	Support cross-regional involvement of departments or rescue units and document their activities. A crisis could require different groups of helps from different regions and geographical places.
Scenario	Without waiting for central coordination orders, first intervention units are sent to start extinguishing the fire, provide security, and to provide more accurate view and information of the situation.
Context	
Priority	High
Source	ACM description and sample scenarios ¹
Issue/Notes	measure of success: send situational information such as the number of dead or injured people, damage to the plane to the central crisis management unit within reasonable amount of time. It could be hours, minutes, or days.

Table 7.4: Structured documentation for ACM's goal with ID:ACMG3

Step 5: Refine goal links

This step refines the links between the goals, to develop the structure and relationships between the goals. Figure 7.3 presents this refinement for the first iteration. The notation is borrowed from the KAOS tool (Objectiver, 2010). Having a big picture of the relationships between goals gives the designers and stakeholders a high-level, graphical view on the system's values, by observing its goal breakdown and the relationships between goal structure elements (e.g. goals, agents). For more detailed information one can refer to goals and links documentation in Chapter 3.

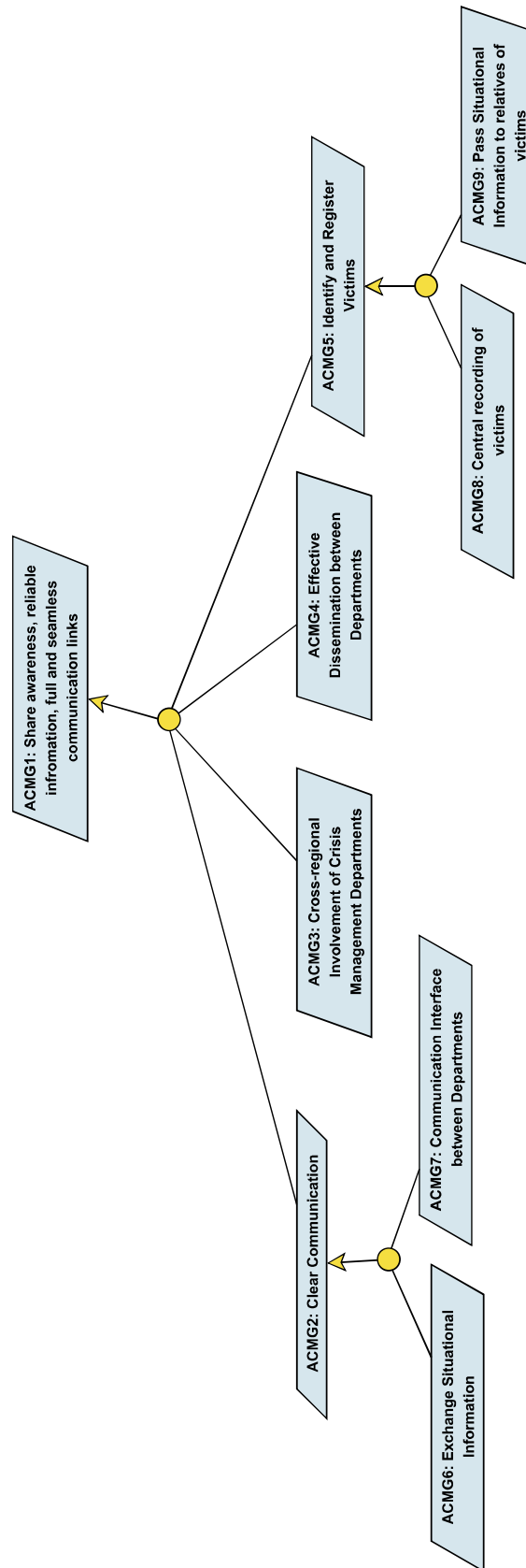


Figure 7.3: Goal Refinement for ACM goals in the first iteration.

Step 6: Document links

“The goal model records the refinement links between goals. In KAOS, a link is documented with details of the refinement that has been made: Name, SysRef, Status and Tactic. In KAOS, the name feature is used to remove any ambiguity. System reference (SysRef) refers to system-to-be or system-as-is. Status records whether the goal is still under refinement. Tactic records the refinement tactic used to derive the sub-goal. In KAOS- β , the tactic feature is used (additionally) to provide source information, which documents or other sources were used in arriving at the refinement. This supports traceability, and also helps in evaluating the goal model” (Tabatabaie et al., 2010b).

Table 7.5 and Table 7.6 show filled forms for the links observed in Figure 7.3. The documentation for the three top goals can be seen in *Step 4*. The link documentation for the remaining goals is presented in Appendix A.2.

ID	ACML1-2
Name	Link from ACMG1 to ACMG2
SysRef	Sys-to-be
Status	Goal under further refinement
Tactic	Designer’s assumption is based on document ¹

Table 7.5: Documentation for the link between ACMG1 and ACMG2

ID	ACML1-3
Name	Link from ACMG1 to ACMG3
SysRef	Sys-to-be
Status	Goal under further refinement
Tactic	Designer’s assumption is based on document ¹

Table 7.6: Documentation for the link between ACMG1 and ACMG3

Step 7: Identify agents

“An agent is an active system component playing a specific role in goal satisfaction” (van Lamsweerde, 2009, p. 260). Analysis of the ACM goals and documents suggests four agents for the start. Goals ACMG6 (Exchange Situational Information between departments and rescue units) suggests the **Departments** as one group of agents. ACMG7 and ACMG8 suggest IT consultants and solution providers as one group of agents. The interfaces of the departments including IT, human, and

other device interfaces are suggested by ACMG3 and ACMG4. These goals also suggest human representatives of different departments and stakeholders or society members that are affected by the crisis as one group of agents.

- Departments
- IT Solution
- Departments' interfaces
- Representatives of departments and stakeholders

Step 8: Link goals and agents

After identifying the goals and agents, step 8 guides the connection of goals and agents, as summarised in Table 7.7. Figure 7.4 displays the graphical presentation of the links between agents and goals.

Agent	Goal
Departments	ACMG6
IT Solution	ACMG7 , ACMG8
Departments' inter- faces	ACMG3 , ACMG4
Representative	ACMG9

Table 7.7: Goals that indicate the existence of agents.

Step 9: Identify obstacles, threats and conflicts

In this step, the designers search for obstacles, threats, and conflicts. The results could affect the goal structure, hence an appropriate iteration to step 3 is considered for this phase. The analysis illustrates that no conflicts between goals have been detected in the first iteration. There are security threats in exchanging situation information and communication interfaces. Also there are threats in exchanging reliably the right amount of information between departments for dissemination and to the relatives of victims and other groups such as media. These threats and conflicts should be highlighted in the architecture design and document and be considered in the implementation phase.

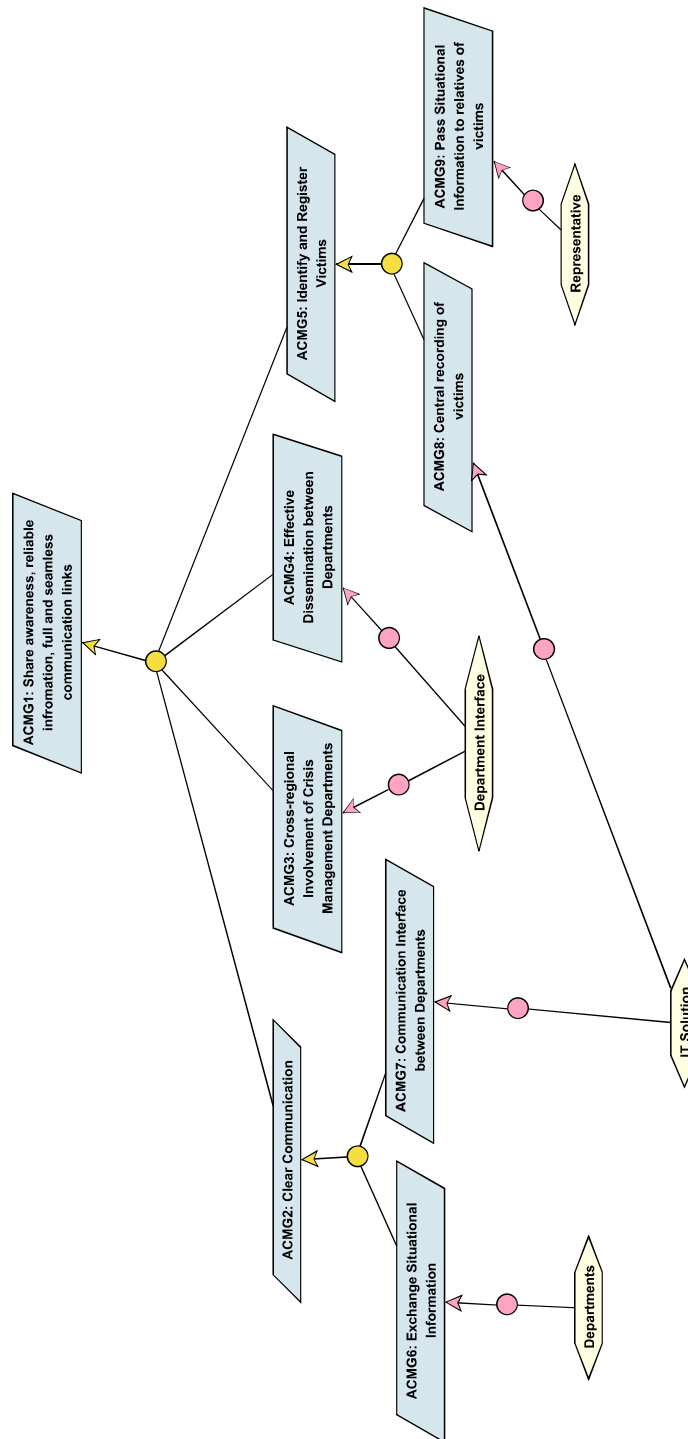


Figure 7.4: The presentation of the links between goals and agents during the first iteration.

7.2.2 Second Iteration

Several iterations through the KAOS- β steps help to collect a suitable amount of information for the designers to start the architecture design process. However, each iteration is time consuming and costly. Therefore, the number of iterations depends on resources available to the EIS development team. To be confident that enough information is collected for this example, we run a second iteration. This helps to collect more data, evaluate the results of the first iteration, and increase an assurance of the final results.

Figure 7.5 presents the results of the second goal refinement. Compared to the results of the first iteration (Figure 7.3) detailed goals are introduced to present different aspects of the top-level goals. There is a fine line between defining the detailed goals and the requirements, hence extra care is applied to not define any requirements in this iteration.

Figure 7.6 presents the links between the goals and agents in the second iteration. This figure includes the presentation of the results for steps 7 and 8. The conflicts which are the results of step 9 is presented in Figure 7.7. In this figure the small red lightning on the links between some of the goals presents the conflicts. The results of the second iteration developed enough confidence to have enough information to start the next phase of the method, applying EAPM.

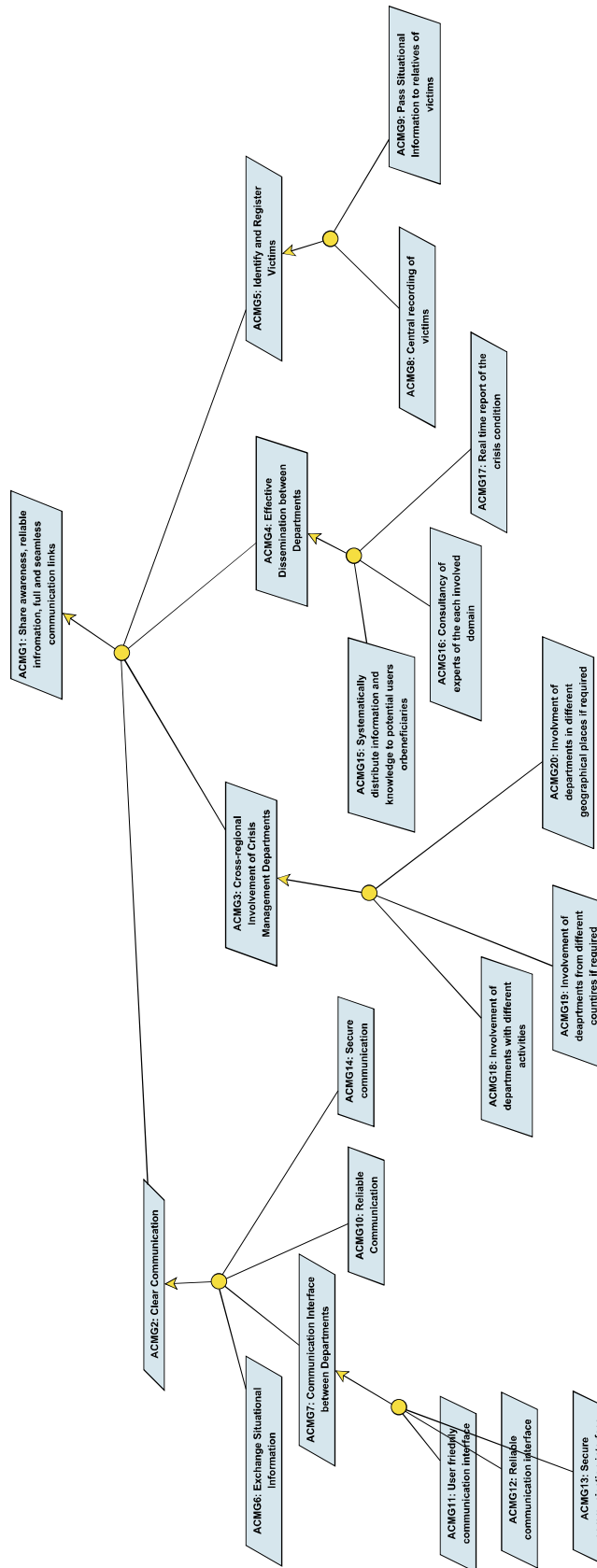


Figure 7.5: Second iteration of goal refinement for SCM goals. The parallelogram presents the goals.

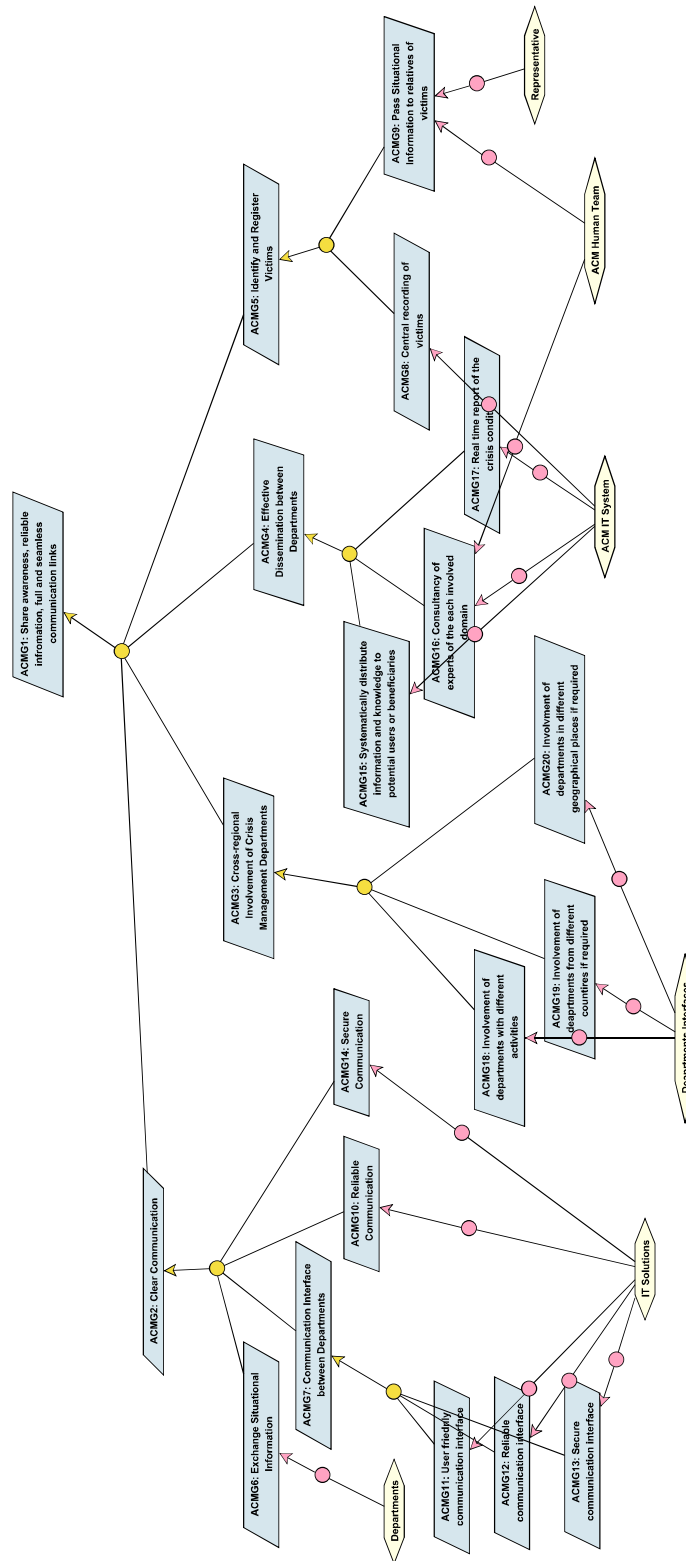


Figure 7.6: Links between the goals and agents in the second iteration. The yellow hexagon presents the agents.

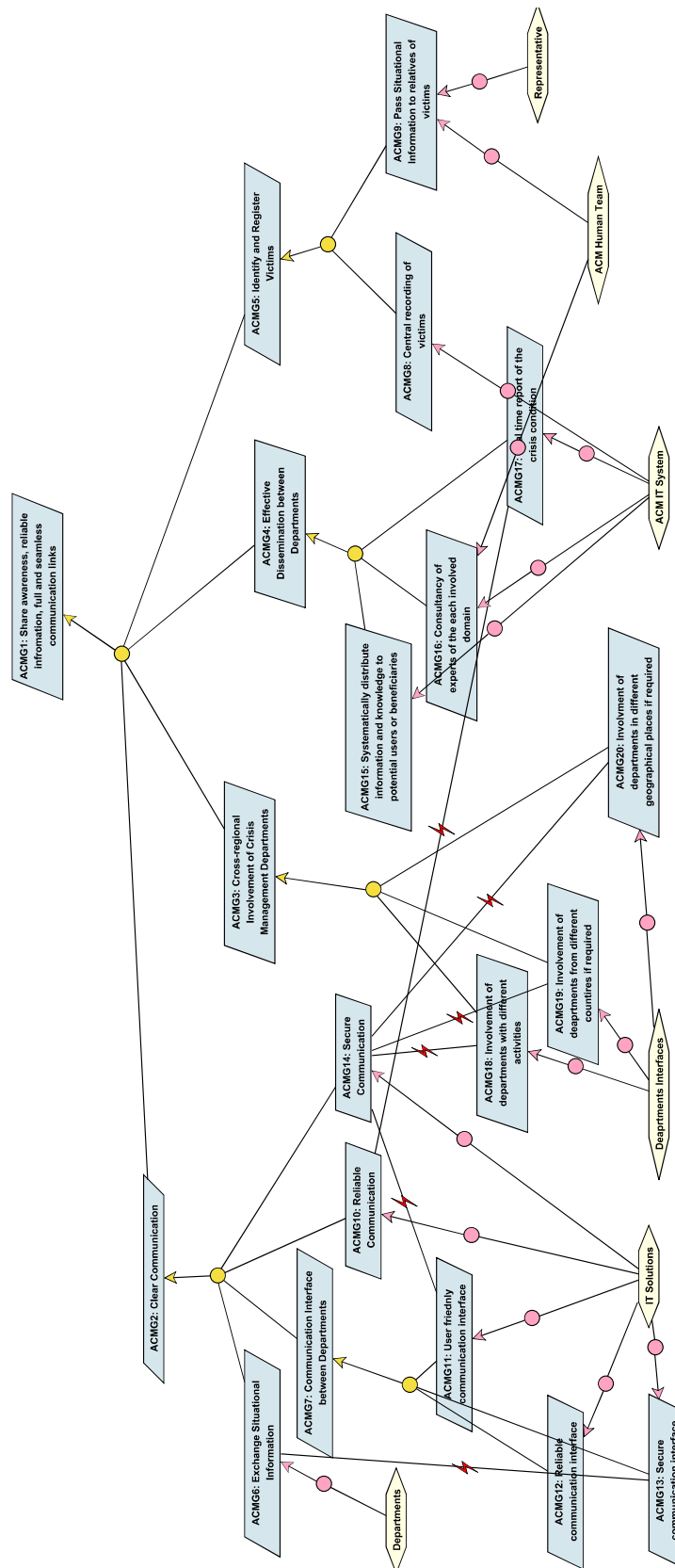


Figure 7.7: Conflicts between goals in the second iteration. The red lightning on the links between the goals symbolises the conflicts between those goals.

7.3 Applying EAPM

EAPM creates a transition from a goal structure to starting point of EIS architecture design. The aim is to provide suitable information for EIS architects to start the process of designing an EIS architecture, regardless of the techniques or strategies they wish to adopt. This section presents the results of applying EAPM to the output of KAOS- β for ACM. Parts of these results is captured in Table 7.8 to Table 7.11; the rest of the forms are located in Appendix A. Data analysis on the output of KAOS- β and ACM documents indicates eleven quality attribute for the EIS architecture. For each of these quality attributes a form is used to document the required information. The elements of these forms are filled with the information gathered from each of EAPM's steps. Thus each form presents the output of all the steps of EAPM.

Attribute (s)	Reliable Communication
Scenario (s)	After an alarm has been triggered, critical information (smoke location estimate and composition) needs to be sent urgently to the appropriate persons (e.g., the fire department) for analysis and decision on how to limit the potential impact.
Goal (s)	ACMG10
Environment	At the start of activating the crisis management system
Stimulus	Critical information send to the system
Response	Distribute the information to the valid departments
Measure	Valid departments and people receive the information within the specific period of time (with no delay)
Strategy	Process group view ³
Reasoning	“This strategy informs operational group members when another member fails, recovers, joins, or withdraws voluntarily, or when some other change to a global property of the group occurs” (Birman & Joseph, 1987, p. 5)
Architectural diagram	

Table 7.8: Architectural information for Reliable Communication quality attribute.

Table 7.8 presents the information regarding to reliable communication. A scenario is defined to justify the existence of this quality attribute. In addition, this scenario could be used in the later phases of evaluating the EIS architecture and even final product testing. The goal that indicates

this quality attribute is presented in the Goal criterion (refer to ACMG10 in Section 7.2.1). The Environment is filled with the information about the environment of the EIS, if it is running under normal condition or the quality attribute affects particular phase or aspect of the EIS. In the case of **reliable communication**, this quality attribute has an strong impact in the early phases of using CMS, including when the system is launched by the users. Based on the information gathered from the scenario, the remaining criteria are completed. A similar approach is followed to fill the forms with the information for the rest of quality attributes.

Based on the information collected by EAPM, an EIS architecture for ACM is designed. The ACM documentation analysis and the scenarios shaped a business process diagram that is presented in Figure 7.8. This diagram consists of five major tasks. Each task has a number of inputs and outputs. For example, in the start of using the ACM users insert the connection information (e.g. username, password, and position). Task 1 processes the entry data and if the suitable information is provided a connection will establish. After creating the business process diagram, more layers of the architecture that aim to address the business process are built up based on the chosen strategy and style (e.g 3-tier style (Rozanski & Woods, 2005)). For the EIS architecture, this business layer will link to IT strategic layer. Figure 7.9 demonstrates the links between business and strategic architecture layer. The aim of designing an EIS architecture for ACM is to demonstrate one possible EIS architecture and not necessarily the best one, therefore no further evaluation for the architecture is carried out.

Attribute (s)	Safe Communication
Scenario (s)	The system will make all or part of the information available to the different groups of stakeholders according to their access rights. Also the system is under the control of safety standards.
Goal (s)	ACMG14
Environment	Distributing the information under safety standards
Stimulus	Distribute the information between stakeholders
Response	follow the safety standard criteria
Measure	Information does not pass to non-valid stakeholders
Strategy	Strict access list
Reasoning	Function and level of access are assigned to roles
Architectural diagram	

Table 7.9: Architectural information for Safe Communication quality attribute.

Attribute (s)	Reliable DB
Scenario (s)	Different types of information about the crisis including the position and movements of the victims are inserted in database(s) and stakeholders can have different level of access to the detail information with appropriate properties
Goal (s)	ACMG8, ACMG4
Environment	Crisis management system is storing and retrieving information
Stimulus	Information is inserted insider database (s)
Response	Save and become available to the other functions to be retrieved
Measure	Information save and retrieved in real time into and from database (s)
Strategy	Oracle Database Lite Client
Reasoning	<ul style="list-style-type: none"> • Secure data access from mobile devicesSmall footprint database enables secure offline access to your corporate data at any time from your mobile device • Access corporate data on the roadChanges made offline in Oracle Lite are tracked and can later synchronized with the backend Oracle Database using the Oracle Database Lite Mobile Server • Broad platform supportSupport for Windows 2003/XP/Vista, Redhat Linux, Windows Mobile 5 & 6, Symbian 7, 8, & 9, and embedded Linux for SH4 and xScale
Architectural diagram	

Table 7.10: Architectural information for Reliable DB quality attribute.

Attribute (s)	Usability
Scenario (s)	Different stakeholders send and request information and the system provides appropriate feedback and assistance within specific period of time (real time).
Goal (s)	ACMG11
Environment	System respond to the requests from stakeholders and other systems.
Stimulus	Stakeholders or other systems send or request information
Response	System send the suitable amount of information within specific time
Measure	
Strategy	Oracle Loyalty Analytics (Oracle, 2011a)
Reasoning	<ul style="list-style-type: none"> • Understand users transaction trends and response accordingly . • Distributed valuable information to suitable users.
Architectural diagram	

Table 7.11: Architectural information for Usability quality attribute.

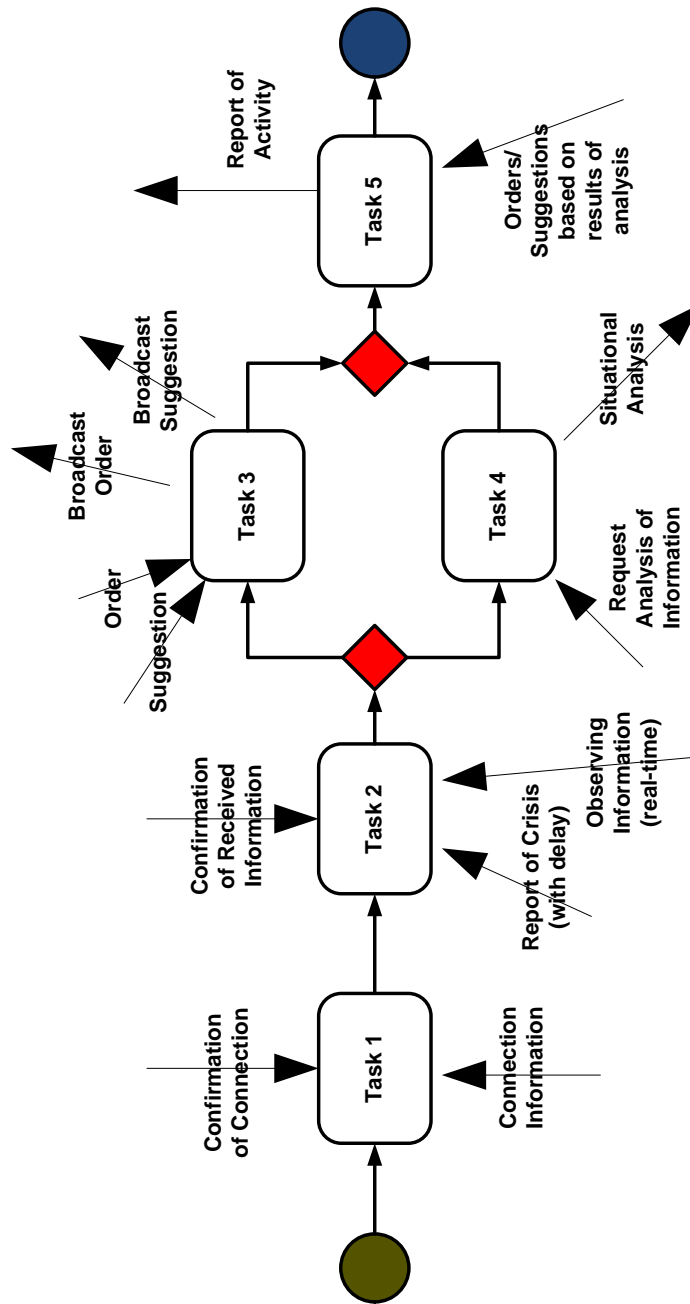


Figure 7.8: A business process diagram for a scenario of Airport Crisis Management. This business process is presented as the top layer in Figure 7.9

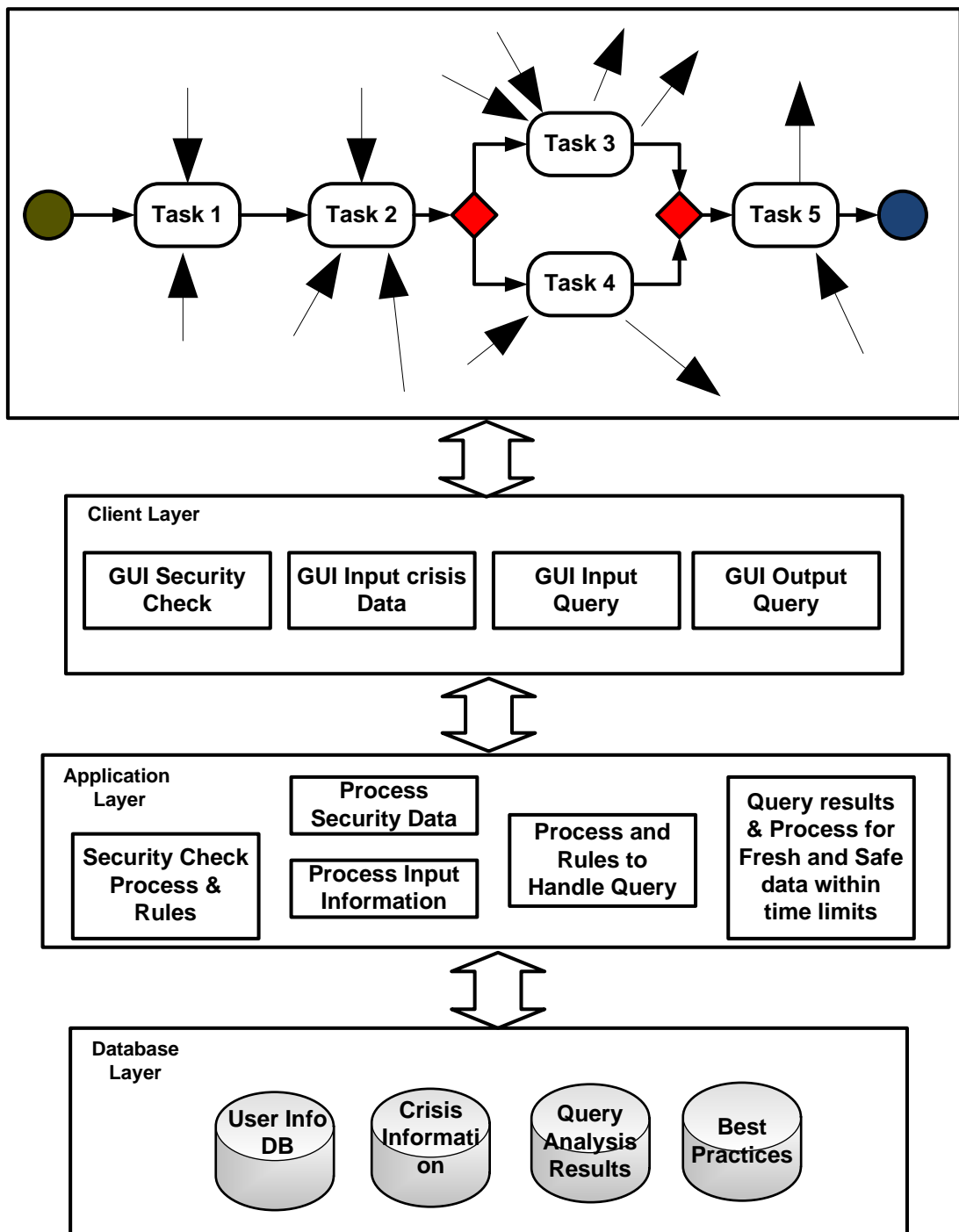


Figure 7.9: Partial EIS architecture for ACM using three-tiered architecture style.

7.4 Conclusion

This chapter presents the results of applying the method to a second example of EIS to test applicability. The second example, ACM, in addition to processing the characteristics of an EIS, has similarities with the earlier EIS example, stroke care.

Applying these approaches in a different EIS example with similar characteristics developed an opportunity to determine that the method is not specific to one particular example of EIS.

Indeed, the contributed parts of the method, namely KAOS- β and EAPM may become mature by applying them to more examples of EIS.

Ideally, future application of the method will be carried out by other researchers and developers, which will help assess the learnability of the method.

The next chapter presents a detailed qualitative evaluation of the method, using the results of the two case studies.

Chapter 8

Evaluation

In the past, expert review has been the main technique used for evaluating processes and process models (Feiler & Humphrey, 1993). However, access to experts is usually limited and costly, hence other solutions are required to make evaluation feasible. Such techniques should allow designers and developers to review the steps of a process in early phases and then pass the reviewed process to the domain experts for further and complete evaluation. Evaluating the process model in advance to presenting it to the domain experts does not prevent objective evaluation.

The objective of this chapter is to present and use different approaches to evaluate a process model without the use of process experts. Indeed these methods are for evaluating the process models and not the results of applying them to a certain domain. To evaluate the results, domain experts provide invaluable knowledge.

This chapter includes two main sections; Section 8.1 presents the evaluation process and results of KAOS- β and Section 8.2 presents the evaluation process and results of EAPM. Different evaluations for process models are presented and demonstrated in these two sections.

8.1 Evaluation of KAOS- β

The KAOS- β process model is an extension and reordering of KAOS's heuristic rules, adapted to EIS goals. Evaluation is a non-trivial problem, particularly in the context of a small project that does not have the capacity to apply the process in many realistic situations. There is little guidance on process evaluation, and that which exists is often inapplicable. For example, (Gruhn, 1991) proposes an approach that evaluates application of a process in a specific context; the approach also focuses on temporal and dynamic aspects of

process application: neither is appropriate here.

KAOS- β is evaluated in three ways: appeal to standards, internal validity (or soundness), and external validity (based on (Curtis, Kellner, & Over, 1992)).

The appeal to standards is made by specifying the KAOS- β process model using a standard notation for process models. Object Management Group (OMG) defines a set of UML stereotypes for defining processes and their components using the SPEM – Software & Systems Process Engineering Metamodel Specification (SPEM, 2009).

The OMG defines SPEM as a standalone meta-model built upon UML 1.4 and UML profile (Object Management Group, 2008). SPEM “is used to define software and systems development processes and their components” (Object Management Group, 2008, p. 20). SPEM first appeared in 2002 and has been developed by process engineers; it is internally grounded and limited to the minimum elements necessary to define any software and system development process, without adding specific features for particular development domains or disciplines (Object Management Group, 2008). SPEM merges different characteristics of a process such as process structure, behaviour, and content.

To demonstrate SPEM in practice, Figure 8.1 and 8.2 present detailed activity and workflow diagrams for an imaginary case designed in SPEM. In addition to graphical notation, the standard defines rules and constraints for processes, using the Object Constraint Language (OCL). SPEM provides the ability to map between the content of an activity diagram, a process and a project plan; hence this “breakdown structure provides key information attributes that provide the project planner with the right guidance to make these instantiation decisions” (Object Management Group, 2008, p. 166).

Designers use SPEM because it provides the necessary concepts for modelling, documenting, presenting, managing, interchanging, and enacting development methods and processes.

Specifying KAOS- β with SPEM improves confidence in the validity of the process model. To further improve confidence, web-based tool support for KAOS- β is implemented in the Eclipse Process Framework (EPF). EPF is an open source tool platform for process engineers and project managers to author, tailor, and publish method and processes (Haumer, 2011); it is based on the latest version of SPEM. EPF can be used without full knowledge of Eclipse, and provides a process editor that supports different breakdown structure views and graphical process presentations (Haumer, 2011): this makes it accessible to non-expert users.

The result of using EPF to produce tool support is a static website and an

XML description of the process model. The benefit of using this framework is to apply the SPEM standard easier and to produce a free tool that can be sent to the users and other designers to be used and evaluated. The EPF tool supporting KAOS- β is called Kestrel. A snapshot is shown in Figure 8.3.

The specification of KAOS- β in SPEM and EPF identified several omissions from the manually developed version. SPEM prompted addition of roles, and structuring of tasks: the final version comprises four roles, responsible for, modifying, or performing thirteen distinct tasks (tasks include a number of steps). SPEM also use streamlined the process, so that the outputs of one task form the inputs to the next task. The EPF tool use led to clear templates for documenting goals, tasks, and agents.

Evaluating KAOS- β using a process model standard clarified the importance of defining elements for process models explicitly. The final version includes all the process model elements requested by the standard. KAOS- β , as other process models, could evolve with feedback from users and additional documentation. As this is a trial version, and because of the limitations in time and access to the domain users, this aspect of the evaluation is considered as complete at this stage and the evaluation continues in the other directions.

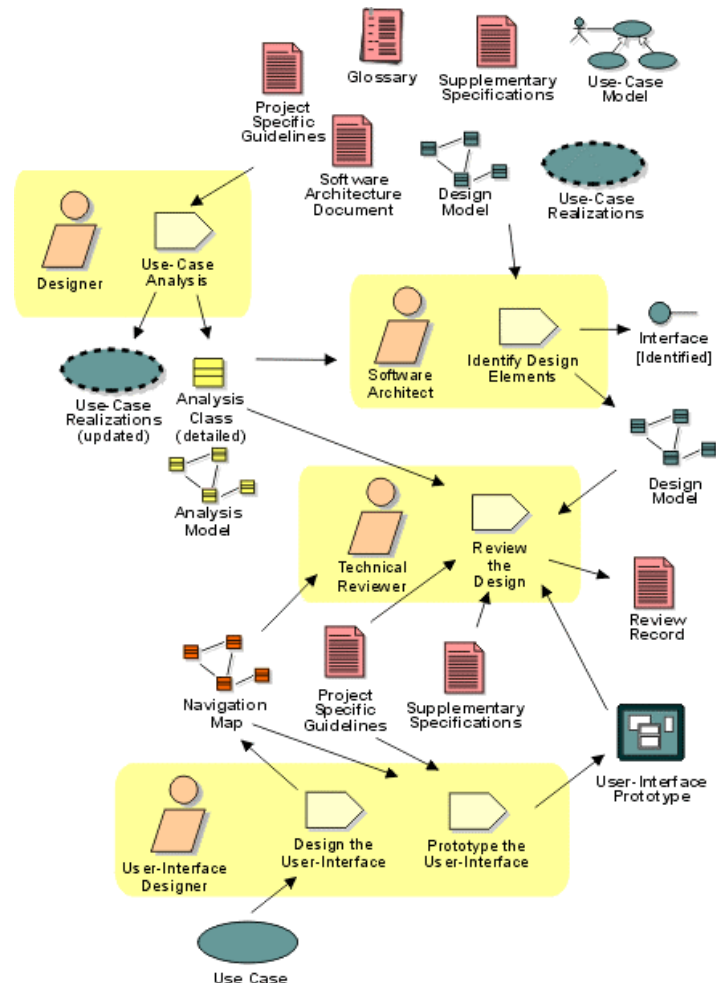


Figure 8.1: A detailed activity diagram designed using SPEM: from (Object Management Group, 2008, p. 162)

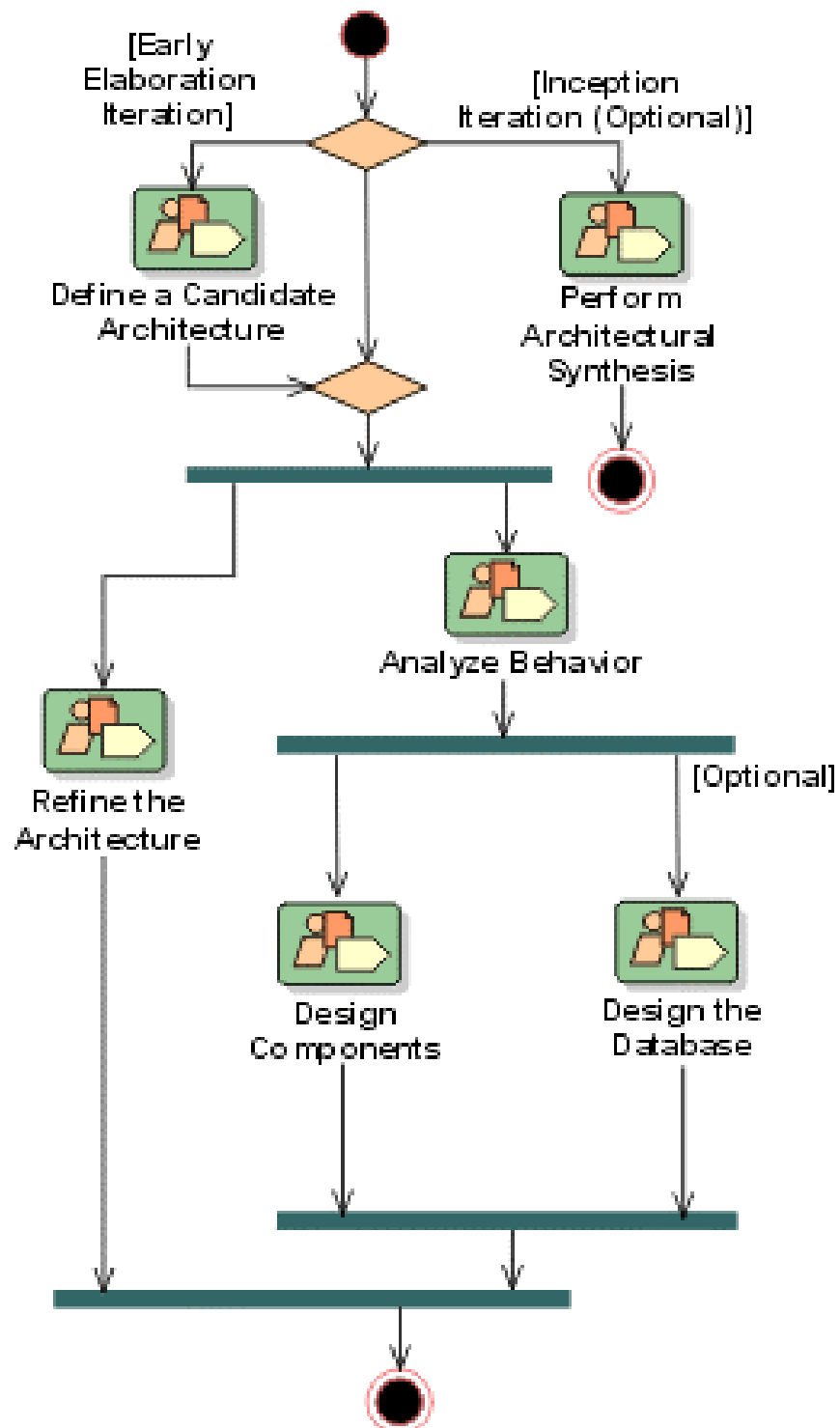


Figure 8.2: A workflow diagram designed using SPEM, from (Object Management Group, 2008, p. 161)

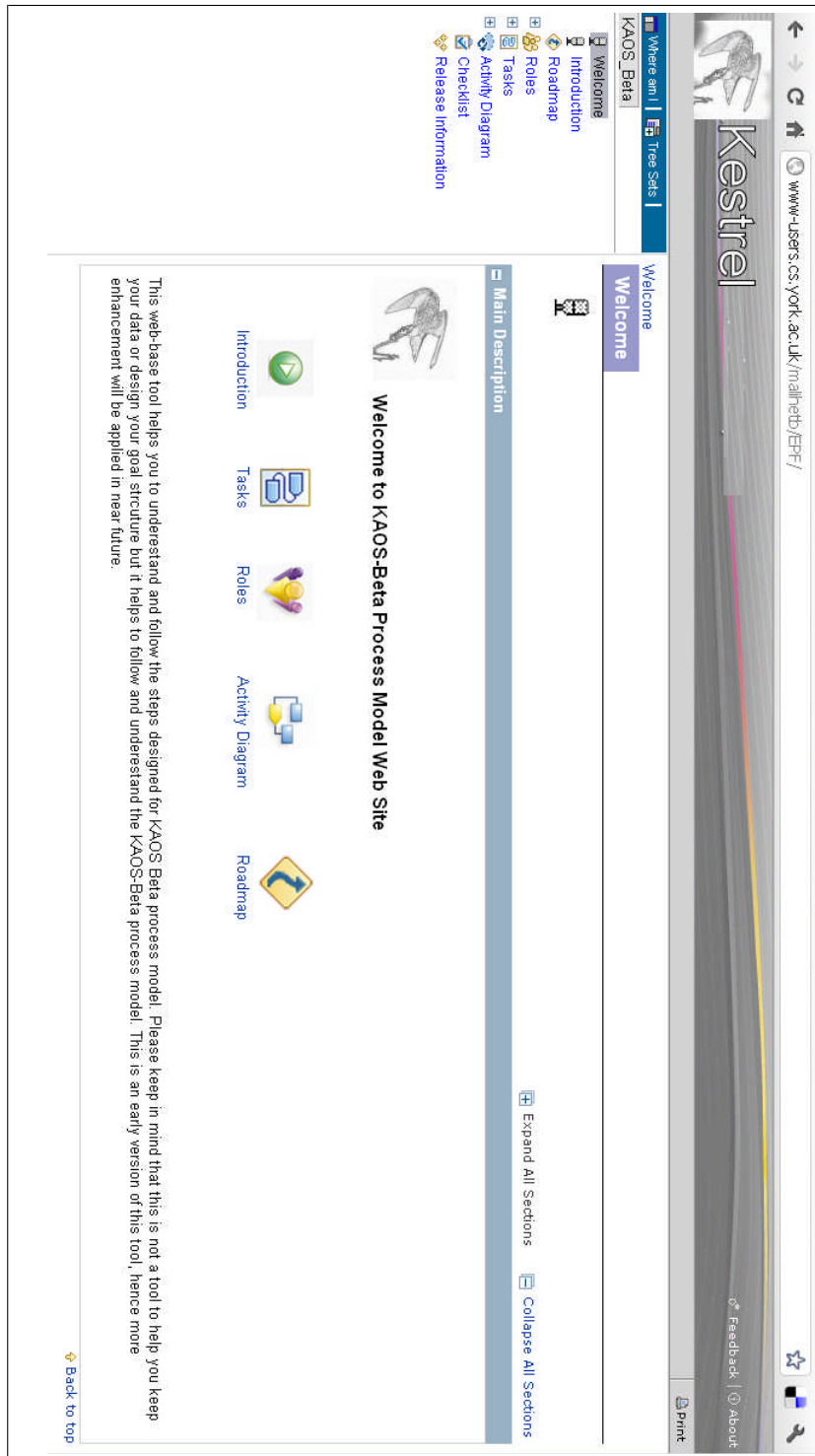


Figure 8.3: A snapshot of Kestrel, the tool support for KAOS- β

The second part of the evaluation of KAOS- β considers internal validity. Internal validity addresses the detail of the process: the structure, applicability and heuristics, as well as how the outputs are validated. The KAOS- β process model is represented as a series of tasks and steps. Clearly, identifying a process, its steps and links makes it possible to address its internal validity. It can be seen that the process and the steps themselves conform to most of the process characteristics (i.e. role, task); KAOS- β also offers the potential for evolvability and adaptability. KAOS itself is a widely-used approach to requirements engineering, and we can assume that it is internally valid. We can thus infer at least some internal validity for the KAOS- β process model through its derivation from KAOS. The internal differences between KAOS- β and KAOS are derived by application of the process to an EIS. Here, internal validity can be shown by appeal to best practice: modularity is added to manage the scope of EIS goal analysis; traceability is added so that goal models can be maintained and can be used to initiate specific systems projects in the enterprise.

The KAOS- β process includes explicit iteration, to support the exploratory nature of goal analysis. The validity of the output of the process (the goal model) is improved through iterative development and oversight of the process and results by the enterprise stakeholders.

In terms of goal evaluation, KAOS- β Step 9 (identifying obstacles and threats) requires the analyst to evaluate the goals by considering interaction and negative aspects of goals. KAOS goes further, for example looking at the converse of goals; KAOS- β evolution should consider adding further goal evaluation heuristics as optional steps, or adaptation options for use in other EIS situations.

KAOS heuristics advise continuing goal identification “to the system boundary” as a stopping condition, and also advise that refinement should continue until each goal maps to one agent. For KAOS- β , we propose a validation check that defines the boundaries of each module and of the EIS (i.e. the scope of the goal analysis), and determines that the identified goals reach the boundaries: this is a coverage condition. The second validation check is that all goals should be mapped to agents – though we allow goals to map to many agents, as described earlier: when the enterprise wishes to proceed to system development, the goal model would form the starting point for requirements engineering, in which lower level goals would be desired and mapped to single agents.

The final part of evaluation considers external validity. These outline the general criteria for a valid process. The validity of the process depends on the qualitative argument on how a process addresses these criteria. If a process

contains detail information such as the time to fulfill a task, quantitative criteria or measurable values could be added to the external validity. At this level a number of criteria that has been investigated by Ramsin (2006) work is used. These criteria are for evaluating software engineering processes and address the general and quantitative aspects of KAOS- β .

To start this phase of evaluation, first the general requirements for a (software engineering) process is considered. We draw on evaluations of process presentation techniques for UML (Dumas & ter Hofstede, 2001) and Petri-nets (Curtis et al., 1992; Murata, 1989), and a general definition of process as a set of activities, associated results and a product (Sommerville, 2007), and add to the requirements of a process the need for defined inputs, defined outputs, and linkage between steps. The type and domain of input and output of the process shall be defined for the users of the process.

The KAOS- β process conforms to these general process characteristics: it is a process to create an enterprise-level goal structure and EIS architecture; it comprises a set of well-defined activities, represented as steps, with associated results that link the activities. As a result, a goal structure which is part of the software product design is created.

Good engineering practice also proposes that a process should be customisable, both in its ability to evolve, and in relation to adaptation to different situations. This cannot be assessed directly at this stage.

The criteria-based evaluation applied all of Ramsin's criteria (Ramsin, 2006); these criteria are summarised in table 8.1. Three criteria require comment: (a) consideration of the clarity, rationality and consistency of the process definition, which cannot be assessed until KAOS- β is fully reported in practice and tested by the domain users; (b) coverage of generic lifecycle development activities is addressed in so far as is possible, but KAOS- β precedes most conventional lifecycle development activities; (c) support for umbrella activities (risk management, project management and quality assurance) have not yet been addressed. The other criteria are addressed and can be seen in table 8.1.

This qualitative analysis illustrates that KAOS- β could be applied for some cases of EIS. We could get confidence about it by testing it in different cases. We believe that there is no one good process that could be applied to all types of EIS, hence instead we should focus on the best practices.

Ramsin Criteria	✓	Comments
Seamlessness and smoothness of transition between phases, stages and activities	✓	KAOS- β retains and improves the flow between activities (steps), facilitating iteration or omission of optional steps
Basis in the requirements	✓	KAOS- β address the needs of the example EIS goal analysis and is aligned with general definitions of process and of EIS
Testability and Tangibility of artefacts, and traceability to requirements	✓	Artefacts are tangible: goals, agents, refinement (goal-model); testability is heuristic; KAOS- β traceability is explicit
Encouragement of active user involvement	✓	Intended and facilitated in KAOS- β , as in KAOS
Practicability and practicality	(✓)	KAOS- β is applicable EIS; it is a practical modification of KAOS. Full practicality needs umbrella activities (further work)
Manageability of complexity	✓	KAOS- β 's clear activities and links minimize complexity
Extensibility / Configurability / Flexibility / Scalability	✓	Addressed by iterative process, modularity; process potentially customisable
Application scope (Information Systems)	✓	Scope is EIS goals/architecture

Table 8.1: Evaluation of KAOS- β against Ramsin's criteria (Ramsin, 2006)

8.2 Evaluation of EAPM

Analytical evaluation is evaluating the subject by dividing it into elements and basic principals (TheFreeDictionary, 2011). Analytical evaluation of the EAPM focuses on the standard basic elements of a process model (i.e. tasks and roles) as well as addressing how practical the process model is, compared to similar architectural process models.

To evaluate the basic elements of a process a review on the qualitative technique has been conducted. The results illustrates the existence of attempts to measure the values of the basic principles. For example, counting the number of roles or tasks and comparing these values of the processes (Garcia et al., 2006; Aguilar et al., 2006). However, there is no evidence of how these measured criteria and values leads to evaluating the quality of a process. For example, if the number of tasks in a process model ‘A’ is less than the number of tasks in another process model ‘B’, does it mean process model ‘A’ is less complicated than ‘B’?

Therefore, the analytical evaluation in this thesis is based on qualitative arguments. The main reason to choose qualitative techniques over quantitative techniques is that process models in general, and in particular the ones studied in this thesis, heavily depend on how they are used. No general quantitative measurement techniques have been found that can be applied to the use of EAPM.

Even though a qualitative approach is considered for evaluating EAPM, there is little research on how to do such an evaluation. In this thesis, two approaches for evaluating EAPM have been used: appeal to standards and appeal to similar process models that are used in practice. The following sections present the results of evaluating EAPM using these two approaches.

8.2.1 Process Model Standards

The first approach to evaluate EAPM is to appeal to standards by specifying this process model using a standard notation for process models. The standard notation, SPEM, was introduced in Section 8.1. This is a general standard for software engineering process models, therefore, it is applicable both in the case of KAOS- β and EAPM.

Figure 8.4 presents the activity diagram using SPEM syntax and axioms (established principles that are applied by OCL for well-formedness check). Activity diagram in SPEM includes a number of steps and the relationships between them. SPEM’s activity diagram follows the general rules of UML activity diagram (Stevens & Pooley, 1999), with small modification in the graphical syntax (SPEM, 2009). Activity diagram in Figure 8.6 is in paral-

lel with the activity diagram presented in Figure 3.5. No additional steps are detected in designing the SPEM activity diagram, and no new phase is suggested by SPEM to be added. However, SPEM documents suggest the concept of Role as an important element of any process and process model and, as no roles can be defined in SPEM's activity diagram, a use case diagram is designed to cover the interacting roles (see Figure 8.5).

Defining EAPM using SPEM does not reveal any conflict between EAPM and the process model standard but has helped to define additional elements. In addition EAPM is defined systematically by SPEM standards using EPF.

Originally one role, an **Architect**, was identified by EAPM for the Stroke Care example. Applying SPEM and investigation of the possible roles for designing the use case diagram, identifies two more additional roles: **Domain Expert** and **IT/Domain Expert**. Each of these roles provide their view point. analysing their view point provide additional knowledge of the enterprise for the architecture, and using this knowledge an architect could allocate the tasks to the suitable role.

Eclipse Process Framework (EPF), is an open source plug-in that helps designers to define their processes systematically and according to SPEM standards. The output is an automatically published web-based tool that can be easily used to support the processes. A snapshot of the output is presented in Figure 8.6¹.

The tool developed using EPF for EAPM is EAPM-Tool. Figure 8.6 presents all the tasks and roles defined for EAPM; seven tasks for the seven steps of the EAPM is defined in EPF. Three roles responsible for these seven tasks are defined, as explained above.

To determine the architectural drivers and quality attributes, in addition to the role of architect as a primary performer, a domain expert that has IT knowledge is defined. The IT/domain expert role bridges the gap between the architectural knowledge and enterprise knowledge and would help to define the quality attributes and drivers to addresses the enterprises goals. The two roles of domain expert and IT/domain expert are defined as the result of developing the EAPM-Tool in EPF. Process model developers are required to define the process model steps by step in detail for the users of the process model.

Another advantage of using EPF to design EAPM tool support is defining and justifying the inputs and outputs of the process model's tasks². In the EAPM description (see Chapter 3) tasks are defined but the sequential

¹Note: the template of the tool is similar to Kestrel tool, however, the content is different.

²EPF introduces the tasks and the steps inside tasks.

relationships are implicit. Developing EAPM–Tool forces the process model designer to make the inputs, outputs, and the relationships between the tasks explicit. Explicit inputs and outputs help the users of the process model to clarify their expectations, and helps to evaluate the correctness of applying each task by comparing the results with the expected results³. Explicit relationships help the Process’s users to understand the progress of a process better.

The result of developing the EAPM–Tool is that EAPM describes seven sequential **tasks** that would be performed by three **roles** and thirteen **work products**⁴. This results illustrates that according to SPEM, the first version of EAPM did not define all the elements of a process (i.e. assistant roles and explicit inputs and outputs). However, the current version of EAPM includes these standard process elements.

In summary, the two main issues of developing a process document are lack of visualising a big picture and explosion of details. Researchers narrow down the descriptions to the use of diagrams such as flow charts (Humphrey & Kellner, 1989) to create a visual picture. The aim of these diagrams is to define required elements for a process such as **Roles** and **Tasks**. UML defines a set of stereotypes for defining processes and their components using the SPEM. To facilitate the understanding and use of a process, a web–based tool using EPF is developed. EPF uses the concept of processes and process models used in SPEM. Using EPF to develop a tool to support EAPM lead us to analyse and evaluate the elements of EAPM in detail. This evaluation makes the missing elements clear or it helps to justify them. In addition to this evaluation technique, I considered another approach that is based on comparing EAPM with another process model that shares the same objectives.

³Similar to the concept of black box testing, when the test case does not test the internal structure, but the input and the expected output.

⁴In EPF work products present artefacts and outcomes of tasks

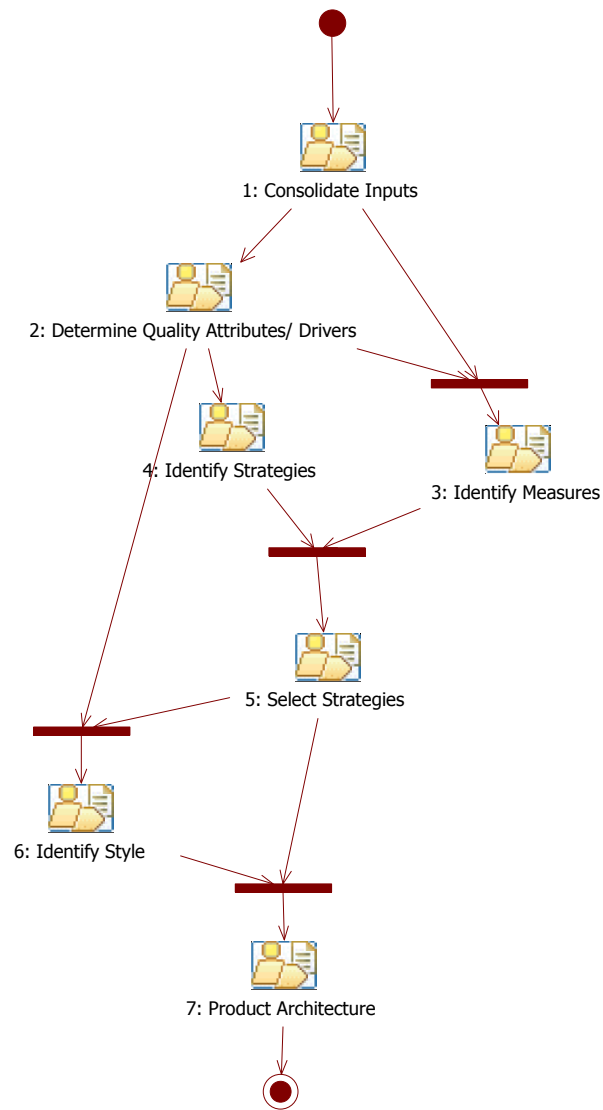


Figure 8.4: Activity diagram for EAPM, designed using SPEM, to present the tasks of EAPM.

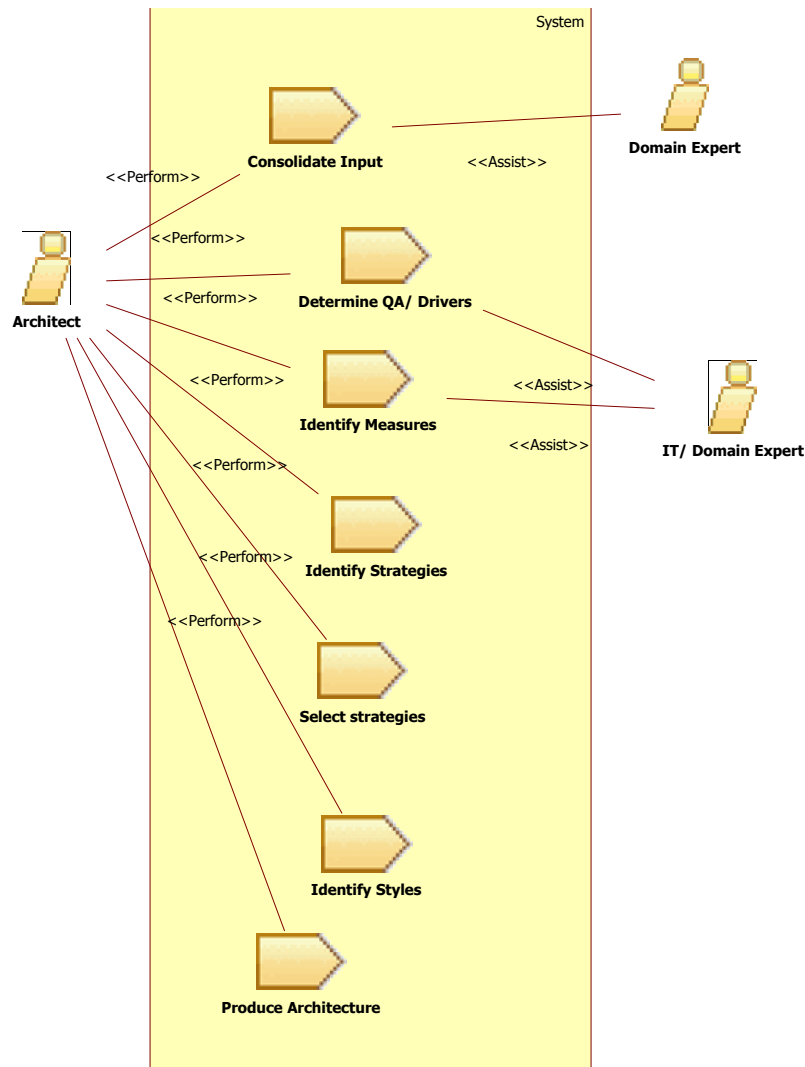






Figure 8.5: Use case diagram for EAPM designed using SPEM to present EAPM roles. Roles in SPEM are presented using  symbol; equivalent symbol for actor in UML is: . Use case in SPEM presented by: , and its equivalent symbol in UML is: .

Eclipse Process Framework Composer

Where am I | Tree Sets | Feedback | About | Print

Task: Architecture_Attribute Determine the architectural drivers/attributes (less than 10) Expand All Sections Collapse All Sections

Purpose Collect architectural information Back to top

Relationships

Roles	Primary Performer:	Additional Performers:
	<ul style="list-style-type: none"> Architect 	<ul style="list-style-type: none"> IT/Domain Expert
Inputs	Mandatory: <ul style="list-style-type: none"> QA Knowledge 	Optional: <ul style="list-style-type: none"> None
Outputs	<ul style="list-style-type: none"> List of Quality Attributes and Drivers 	

Main Description By analysing the collected data from previous task, some architectural information such as quality attribute and drivers is generated. Back to top

new_custom_category

- Consolidate Input
- Architecture_Attribute
- Identify Measures
- Architectural Strategy
- Strategy Combination
- Architectural Style
- Candidate Architecture
- Architect
- Domain Expert
- IT/Domain Expert

Figure 8.6: Screenshot of the EAPM-tool that is developed by EPF to support EAPM.

8.2.2 Process Models Comparison

So far we have used process model standards (e.g. SPEM) as guidance that provide a general roadmap for process model development. This analytical evaluation helps us to identify the missing elements for the EAPM. The next step in evaluating EAPM is to analyse its functionality by comparing it with a valid process models that has a similar objective and approach. “A process model will be called a valid model iff (if and only if) there exists at least one successful process path” (Soffer & Wand, 2004, p. 525); validity of a process is with respect to a given goal.

(Clements & Bass, 2010b) introduce a process model that uses the concept of goals to design a software architecture. They claim this process model has been used successfully, hence this process model has at least one successful process path and could be considered as a valid process.

The PALM (Pedigreed Attribute eLicitation Method) is a method that is piloted in a real-world setting (Clements & Bass, 2010b). PALM uses the concept of goals to design software architecture for large scale and complex systems. The main objectives of using PALM are to empower architects to spot missing requirements and to question difficult requirements that may not be necessary because they do not support any important business goal (Clements & Bass, 2010b). The output of PALM is a “prioritized list of business goals and the associated quality attribute requirements that drive from the stated business goals” (Clements & Bass, 2010b, p. 21). Thus, PALM is aligned with the objectives of EAPM. These process models address the same challenges with a similar approach. This alignment in the objective and approach, motivate the selection of PALM for comparison with EAPM. The methodology used for comparing these process models is criteria-based evaluation. In this method explicit criteria are used as a basis for assessment (The Danish Evaluation Institute, 2004). We investigate how each of these processes could address specific list of criteria. This analysis helps to develop a vision for different aspects of process models.

Clements and Bass (2010b) have evaluated PALM by applying it to real industrial cases and collecting feedback from domain experts (Clements & Bass, 2010b). PALM includes 7 steps that are applied in a 2-day exercise (Clements & Bass, 2010b). Following is the main description that is published to introduce the use of PALM. To estimate the length of running a PALM session, for each step, time is roughly estimated and presented in parenthesis. The information about the estimated time could be provided for EAPM and KAOS- β because there is no access to the industrial cases in the real environment.

1. PALM overview presentation (30 minutes)

2. Business drivers presentation (60 minutes)
3. Architecture drivers presentation (30 minutes)
4. Business goals elicitation exercise (2 hours)
5. Identifying potential quality attributes from business goals (2.5 hours)
6. Assignment of pedigree to existing quality attribute drivers (2.5 hours)
7. Exercise conclusion (30 minutes)

To compare EAPM and PALM a number of criteria as the basis for comparison is required. Humphrey and Kellner (1989) suggests primary objectives and criteria that process models should consider. These criteria are: *enable effective communication*, *enable process reuse*, *enable process evolution*, and *facilitate management of the process*. Ramsin (2006) also identifies a number of criteria for a process model during his research. The criteria that could be applied for general process models and not just specific to software development are: clarity, rationality, accuracy, and consistency of definition; seamlessness and smoothness of transition between phases, stages and activities; basis in the requirements; encouragement of active user involvement; practicability and practicality; manageability of complexity; extensibility, configurability, flexibility, scalability; application scope (Information Systems). In Section 8.1 these criteria have been used to evaluate KAOS- β . Table 8.2 uses the criteria introduced in Section 8.1 to be used as a framework for comparing PALM and EAPM.

The criteria in Table 8.2 as categorised to **Addressed** for criteria that are addressed fully by the process model; **Partly Addressed** for the criteria that are not fully addressed by the process model, perhaps because of lack of public documentation or knowledge about specific process model; **Not Addressed** for the criteria that are not addressed by the process model.

Criteria	PALM	EAPM
Enable process reuse	Addressed	Addressed
Enable effective communication	Partly Addressed	Addressed
Facilitate management of the process	Addressed	Addressed
Support evolution of the process	Addressed	Addressed
Clarity, rationality, accuracy, and consistency of definition	Partly Addressed	Addressed
Seamlessness and smoothness of transition between phases, stages and activities	Addressed	Addressed
Basis in the requirements	Partly Addressed	Partly Addressed
Encouragement of active user involvement	Addressed	Addressed
Manageability of complexity	Addressed	Addressed
Extensibility, Configurability, Flexibility, Scalability	Addressed	Addressed
Application scope	Addressed	Addressed
Practicability and practicality	Partly Addressed (No Evidence)	Partly Addressed

Table 8.2: Results of comparing EAPM and PALM using Ramsin (2006) criteria. The values for the criteria are **Addressed**, **Partly Addressed**, **Not Addressed**. When there is no evidence, PALM developers might partly addressed the practicability and practicality criterion, but we do not have evidence.

Enable process reuse :

Humphrey and Kellner (1989) recommend that a specific software should be able to instantiate and execute in a reliable repeatable fashion across multiple software projects. EAPM and PALM are both process models, hence they are not specified for a specific project (i.e. stroke care). The general nature of process models means that PALM and EAPM can be used for different EIS projects. Process reuse is supported by the objective of structuring the goals, and by associating architectural quality attributes to the goals. The tool developed for EAPM also is not restricted to a project.

Enable effective communication :

“Process models are especially useful for sharing knowledge and expertise” (Humphrey & Kellner, 1989, p. 3). PALM and EAPM are developed in different ways; this influences the effectiveness of communication of the process. PALM is developed by software architecture experts in Carnegie Mellon University. It expresses expert knowledge and has received positive evaluation by industrial experts (Clements & Bass, 2010b). However, it does not provide much practical support for the novice users. The method should be used in sessions run by PALM experts for domain experts. EAPM, by contrast, was developed experimentally, through analysis of an example EIS (the stroke care example). It provides practical examples. The tool support facilitates use of the process, as well as communication between developers and users. Thus, the two process models both provide aspects of effective communication, but EAPM is arguably more accessible to non-expert users.

Facilitate management of the process :

Management of the process can be achieved by facilitating “effective planning, control, and operational management of software processes” (Humphrey & Kellner, 1989, p. 3); this can be accomplished initially through understanding of the process. One concern is that the process models considered here are fully understandable by their originators. PALM, in its first step, provides an introduction to the process model, its elements, and business goals, using natural language. This introduction creates understanding of the PALM process model and consequently facilitates management of the process. On the other hand, EAPM uses a tool as well as text description, to introduce the process and its elements, as well as providing the facility to add detailed information to the process to facilitate planning and control of the pro-

cesses. In practice, if process model experts are available, it is better to have a session to explain the steps and related details; however if the process model experts are not available, then the description and tools should provide the required information. Hence, EAPM and PALM address this criterion using different approaches but they are both flexible enough to allow process users to plan and control the flow of the process model.

Support evolution of the process :

A process model supports the evolution of a process by satisfying the two following objectives: “(1) serving as a storehouse for modifications, lessons learned, and tailoring; and (2) analysing the effectiveness of changes in a laboratory or simulated environment before actually implementing them” (Humphrey & Kellner, 1989, p. 3). PALM and EAPM are potentially able to evolve, to address EIS architecture issues that were not foreseen. Both process models are abstract, reusable, and flexible enough for customisation. However, there is no empirical evidence of evolution at this stage. EAPM is a new process model and there is limited public information on PALM. Hence this criterion is recorded as being addressed partly in both cases.

Clarity, rationality, accuracy, and consistency of definition :

“The methodology should be well-documented (comprehensive, clear, rational, accurate, detailed and consistent description should be provided)” (Ramsin, 2006, p. 201). Ramsin (2006, p. 201) passes two questions: “what should be captured?” and “How?”. EAPM addresses this criterion by providing a text introduction, and an example of implementation. The introduction presents the life cycle and steps of EAPM. In addition a tool is developed according to the specification of EAPM to help the users to understand and apply the process model. However, PALM is designed for brainstorming sessions with the domain experts, therefore the introduction text is very brief and it is not clear whether addresses this criterion fully.

Seamlessness and smoothness of transition between phases stages and activities :

An interpretation of the seamless and smoothness of process models is as follows:

Seamlessness: “To have the seamlessness criterion, there is no costly and error-prone transition between phases, stages, or activities. Gaps between stages which are *impedence mismatches* could be caused by

changes in notations, mindset, and personnel” (Meyer, 2000, p.931). In addition, seamlessness entails direct mapping between the description of the problem and the solution (Meyer, 2000).

Smoothness: The smoothness criterion requires that, to pass from one stage, phase, or activity of the process to the next, without the need to create a new artefact (Ramsin, 2006).

These criteria is satisfied for the EAPM. The interactions between the steps are defined explicitly by defining the input and output artefacts of each task and relating these artefacts for the sequential tasks. As for PALM, the seven steps are high-level and mainly introductory. PALM introduces a sequence of the tasks that use the earlier results. Hence, we found no evidence for lack of these criteria for PALM too, but more detailed knowledge is required to support this claim.

Basis in the requirements :

Ramsin (2006)’s research covers a complete process of software development that starts with requirement elicitation phase and continues to implementation and testing. Ramsin (2006) includes the criterion that a process must base development on the requirements of the project to which it is applied. The requirement in this case refers to the requirements for developing EAPM and PALM. The shared and explicit requirement for both processes is to use the goal knowledge to design the architecture. There are also implicit requirements for modelling a process. For example, we use the SPEM standard for defining the required elements of a process. As most of the requirements are implicit, this criterion is considered partly addressed.

Encouragement of active user involvement :

Ramsin (2006) described user engagement as a vital criterion for activities such as risk management and quality assurance. PALM relies strongly on the involvement of domain experts. This process model is developed based on an assumption that it would be running as a brain storming guidance, hence the encouragement and involvement of users are inevitable (Clements & Bass, 2010b). User engagement continues until the last stage that is collecting feedback from participants. EAPM also encourages active user involvement by involving the domain experts and IT experts with domain knowledge in the early phases. In the later steps in EAPM, the architect plays the primary role however, if the domain expert role still plays a part. Thus both of these process models address active user involvement criteria, with different approaches.

Manageability of complexity :

“The complexity of work-units should be manageable” (Ramsin, 2006, p. 206). There are techniques to limit the complexity of a process however, any technique in designing architecture is very subjective and depends on the project. A process model handles the complexity by explicitly introducing the tasks and other required elements such as roles. Each task of a process could be complex for the users. For example, in PALM, the task of creating a prioritised list of business goals and associated quality attributes could be complex for the users, because there is not enough guidance on how to create this list. In cases it is possible to define a quality attribute directly from a goal. In this case, there is low complexity in the transition from a goal to a quality attribute. If PALM relies on the expertise of architects to define quality attributes directly from goals, then it addresses this criterion. However, EAPM uses scenarios and an explicit data card (Figure 6.15) to document the goals and trace them to the quality attributes. Even though this phase requires the architect expertise to choose the suitable quality attribute that addresses the scenario and goal, EAPM introduces a clear path, which is scenario, between goal and quality attribute to handle part of the complexity of the transition from goals to quality attributes.

Extensibility, Configurability, Flexibility, Scalability :

To apply these criteria to a process model, a minimum characteristics is required (Ramsin, 2006):

- Process model should support projects with different sizes,
- Process model should support projects with different level of criticality,
- Iterative approach usually helps to achieve the above criteria,
- Process model should be flexible enough to be tuned based on the experience gained during the project by developer team.

There is no evidence in EAPM and PALM of anything that would restrict the size, level of criticality, number of iteration, and flexibility of the project. Thus, these process models can be said to address this criteria.

Application scope :

“The application scope depends on the usage context” (Ramsin, 2006,

p. 206). Ramsin (2006) limits the domain of his application to information systems. Limiting the boundaries or defining the domain of applications, processes, techniques, or phases could help the developers to clarify the domain of work and manage the external expectations. PALM is not limited to any specific domain but can be used in designing any software architecture that deals with business goals, by whoever wants to produce a set of requirements or by architects (Clements & Bass, 2010b). EAPM is restricted to EIS architectures, because in this type of system the role of identifying goals has a stronger impact compared to small businesses with clear requirements. However, EAPM could probably be use for other type of software systems where goals can be used for identifying quality attributes and drivers. PALM is used for business-oriented software systems; EAPM is designed for EIS architecture; hence the both address this criterion.

Practicability and practicality :

The review of these two terms in (Ramsin, 2006) leads to the following points that could be applied to evaluate the process:

- the process should be applicable to some groups of projects. The characteristics of these type of projects should be defined,
- it should not be over complex because it will be used less,
- it should avoid the tasks that are distracting from mainstream activities or encumber them from unnecessary details,
- use techniques that focus the development,
- use project management strategies and avoid lack of adequate management measure mainly in large projects with restrictions on time and resources,
- avoid dependency on special tools and techniques.

Both PALM and EAPM apply to software architecture and they follow the software architecture characteristics. PALM evaluated its process model in practice by running a two–days workshop and piloting this process model. The results did not present any exceptional complexity. However, the detailed results of the workshop is not published, hence we cannot analyse the complexity of the PALM steps. On the other hand, EAPM is built to address the complexity of developing an EIS architecture by suggesting goal infrastructure as a starting information. EAPM should not add to the complexity of software architecture but it tries to make the source of quality attributes clear, to

improve traceability. Thus, complexity is partly addressed in EAPM. Both PALM and EAPM are focused on developing software systems; neither of them addresses project management strategies; neither of them is dependent on a specific tool or technique. In short, EAPM and PALM both address some of Ramsin (2006)'s detailed criteria of practicality and practicability.

In summary Section 8.2.2 presents the results of comparing EAPM and PALM. As can be seen in Table 8.2, EAPM could address criteria that are partially addressed in PALM. EAPM does not fully address two criteria: basis in the requirements and practicability and practicality that are partly addressed. However, considering the information in the literature, PALM partly addressed four out of twelve criteria. Based on the results of evaluation, EAPM addresses more criteria with satisfactory compare to PALM.

8.3 Conclusion

Expert review is the main evaluation technique for process models. However, as access to experts is usually limited and costly, other approaches for evaluating these solutions and the results of applying them could enhance the general evaluation process. Thus the evaluation of the method developed in this thesis is organised based on appealing to process models standards, internal validity, external validity, and comparing with other similar process models.

The main lesson learnt from this evaluation process is that the results of process model evaluation are generally qualitative and not quantitative. Recently, some software engineering large-scale process models (e.g. OpenUp) are structured according to this standard. A platform independent tool also has been developed for KAOS- β and EAPM that help not just evaluate the process model but also to distribute it via net. Using this tool the designers share the knowledge about EAPM with the users. The results of analysing the compatibility of EAPM with SPEM illustrate some missing or implicit elements that have been captured and addressed in the revision of EAPM. The results of comparing EAPM with other process model with the same objective and approach shows that PALM addresses fewer criteria than EAPM.

In addition to illustrating the results of evaluating KAOS- β and EAPM, this chapter demonstrates several qualitative approaches to assessing and evaluating process models in general that could provide guidance for future work in developing process models.

Chapter 9

Thesis Conclusion

This thesis focuses on the challenges of the early stages of developing EIS. The EIS development examples and domain analysis illustrates a common gap between goals of an enterprise and developed functionalities of an EIS. This also illustrates lack of shared understanding between EIS developer and enterprise stakeholders and decision makers.

These top-level limitations are the inception for this thesis research questions:

1. What is an EIS and its characteristics?
2. Why might an EIS fail to deliver its functionalities and fail to address stakeholders and enterprise goals?
3. What knowledge is required to understand the required functionalities for developing an EIS?

These research questions are the motivation for further literature and technology analysis. Further literature analysis of technical solutions leads to narrowing down the gaps to the following ones that could fit in the domain of this thesis:

- Lack of explicit process models to identify and structure EIS goals
- Lack of explicit process models to trace and relate the influence of EIS goals on EIS architecture

To address these gaps in this thesis, the following testable hypothesis is defined:

Process models can be developed to provide a precise, repeatable, and documented set of structures for *identifying* and *specifying* EIS goals, and for *relating* EIS goals with a strategic-level architecture.

This hypothesis shapes the top-level goal of this thesis which is:

Process models can be developed to provide a precise, repeatable, and documented set of structures for identifying and specifying EIS goals, and for relating EIS goals with a strategic-level enterprise architecture.

The central contribution of this thesis derived from the thesis hypothesis and is aligned with the process presented in Figure 1.2; the key contribution is as follows:

A novel method that develops goals for Enterprise Information Systems and traces them through to a strategic EIS architecture. This method includes two novel processes, KAOS- β and EAPM.

The two process models that are developed in this thesis address all three criteria mentioned in the hypothesis. KAOS- β and EAPM are precise, because as presented in Chapter 8, they follow the SPEM standard and address the well-formedness rules by developing two web-based tools using EPF.

Where processes are defined for a special case and project, process models are defined to adapt based on the specification of each project and environment. Thus, being repeatable is a characteristic of process models and the main distinction with processes. Therefore, KAOS- β and EAPM, which are process models, are repeatable. To demonstrate repeatability, they have been applied to two independent case studies, from two distinct environments. The results illustrates that considering each project's assumptions, the main backbone of the two process models could be used in both case studies.

This thesis process models are documented using natural language and tool. Different techniques of documentations could help different groups of users to understand and benefit from these process models.

In addition to the main contribution of this thesis, which is developing a method to address this thesis hypothesis; this thesis accomplished following contributions:

EIS definition: Chapter 2 presents the domain analysis of EIS that leads to defining EIS and its characteristics. In Chapter 2, examples of what could be an EIS and what could not be an EIS are discussed.

Modelling Processes: Chapters 3, 4, and 5 of this thesis present the steps and approaches used to develop this thesis method. These modelling approaches can be reused by other practitioners to develop other instances of method, processes, or process models. Best practice in this domain improves the quality of modelling processes.

Process model evaluation: To evaluate KAOS- β and EAPM, Chapter 8 of the thesis identify evaluation technologies that are not heavily dependent on the domain experts. These technologies are applied and tested

for two process models, and no limitations have been identified to limit the range of process models that could benefit from these evaluation techniques.

The next section summarises how these contributions address the research questions posed for this thesis.

9.1 Research Questions Revisited

The thesis is inspired by the challenges and gaps presented in Chapter 2; and accomplished a number of contributions. The following is a summary of the thesis chapters and how the analysis and contributions address the research questions.

9.1.1 What is an EIS and its characteristics?

The main contribution of Chapter 2 of the thesis is to introduce EIS and its characteristics. To answer this question, the analysis of an enterprise and its characteristics is presented. A number of EIS characteristics are summarised in Chapter 2 are as follows:

- Support for business processes
- Support for organisational goals
- Involves and orchestrates multi business processes
- Includes multi partners (Optional)
- Evolutionary development
- High dynamic architecture
- Geographic distribution (Optional)
- Contains sensitive and real-time data and processes
- Flexible: handle changes in business processes and environments
- Open system: interact with other systems (e.g. human, hardware, software systems)

Because there is no standard definition for EIS, the domain and example analysis leads to explicitly defining EIS.

An Enterprise Information System is a software system that integrates the business processes of organisation(s) to improve their functioning.

Examples of what can and can not be an EIS are presented to support the definition. The examples demonstrate the subjective point of view and the gray boundaries of an EIS; in cases, depending on the justification a firm could require an EIS or not. Identifying business processes and the relationships between them helps to determine demand for an EIS. The size of a firm is not a factor for demanding an EIS; the complexity and the relationships between the business processes is a factor.

9.1.2 Why an EIS fails to deliver its functionalities and addresses stakeholders and enterprise's goals?

To address the question of why EIS might fail to deliver its functionalities, Chapter 2 presents an analysis of existing techniques that could be used to identify and structure the EIS stakeholders and enterprise goals. Making the goals explicit improves ability to share knowledge between the EIS developers and the stakeholders. Shared understanding limits unrealistic expectations of EIS functionalities, and leads to development of an EIS that addresses an enterprise's goals.

A number of GOA in the domain of computer science has been reviewed and as the result the first focused gap is identified: *Lack of explicit process model to identify and structure EIS goals.*

This gap motivates the development of a novel process model using the lessons learnt from empirical study of the four GOA.

KAOS- β , the novel process model, is a contribution of this thesis. This process model guides the users to identify and refine the goal structure. This thesis also presents how to systematically develop and evaluate a process model. This learning outcome adds to the knowledge of the process modelling community.

KAOS- β elements, philosophy, and tool are presented in Chapters 3, 4, and 5. Chapter 8 also presents the evaluation techniques and results for KAOS- β .

9.1.3 What knowledge is required to identify the required functionalities for developing an EIS?

From the review of enterprise and software architecture, we propose that the knowledge required to identify required EIS components and functionalities are quality attributes, architectural drivers, and architectural strategies. This thesis's literature analysis of enterprise architecture solutions and the influence of goals on identifying architectural quality attributes leads to identifying this thesis second gap: *Lack of explicit process model to trace and relate the influence of EIS goals on EIS architecture.*

To address this gap and the third research questions, a process model, EAPM, is developed to guide the systematic identification of quality attributes. EAPM uses the goals derived using KAOS- β , and other resources of the enterprise documents. The end point of EAPM is architectural proposal, forming the starting point of EIS development.

9.2 Lessons Learnt

Apart from major contributions outlined above, this research has identified a range of useful insight into EIS, process modelling, and evaluation of process models.

The first lesson is to clarify the definitions, concepts, and characteristics used in a domain. Unclear domain and terminologies leads to creating different understanding of an under question limitation. The domain of software engineering is immense and includes different types of information systems. Each type of information system presents different challenges and development demands. Narrowing down what it means by EIS reveals its specific challenges and helps to focus a research and organises it in a timescale.

A key challenge in EIS research and development is dealing with change. During this research, different approaches and paths of research were investigated. Each solution could lead to identifying more hidden limitations that change the focus of the research. For example, GOA was identified as a solution, this led to more challenges due to the dependency of GOA on experts, lack of clear process models for using a GOA, and a lack of accepted approach to evaluating process models.

Whilst analysing approaches and solutions, assessment of process models became a large part of the thesis. Process models provide a solution for a systematic development, and provide repeatability in the use of approaches. Process models are essential to quality software development and team work, but developing and following process models can be costly, and

has implicit effects on manageability; their importance sometimes has been underestimated (Humphrey, 2007).

While there is a history of studying process models, approaches to developing and analysing process models are limited. Therefore, the analysis of an approach for developing process models and evaluating the results has been a learning curve. One aim is to produce approaches that could be used in the future for developing and analysing other process models.

The next lesson concerns the importance of evaluating results, even if this cannot be done objectively. In this thesis, approaches such as scenario-based and criteria-based evaluations are used for identifying and evaluating a number of EIS goals and process models. These evaluation approaches could benefit the software engineering community and trigger more research in this direction. Another important approach for evaluating the process models and results is the use of domain examples. Suitable EIS examples and case studies could benefit the results considerably, while non suitable examples could damage them. Hence it is important to spend time locating and investigating EIS examples that fit for the purpose of this thesis. This is a challenging task, that can lead to the use of classified and secure data and information that might not be publishable.

The scale of the examples also is an important factor in choosing them. As time and resources are limited, it is important to limit examples in a way that address the main characteristics of this thesis objective but is doable by an individual. Part of rehabilitation phase of stroke care and airport crisis management examples are based on real cases, but have to make realistic assumptions to fit within time constraints and avoid exposing confidential material. Having limited access to the real environment of the examples and their experts makes some of the tasks (e.g identifying and evaluating the goals) challenging and fragile.

A final lesson concerns individual research. Working individually on the examples limits the quality of the results. In practice a team of designers including the domain experts would investigate the domain. Here the developer, the user, and the analyser are all one person. In addition, the lack of a tool to document the results makes the documentation process difficult, and in cases incomplete.

9.3 Future Work

Following is a list of identified future developments for the work presented in this thesis.

- In the thesis two examples of EIS are presented. To become confident

about the results, KAOS- β and EAPM should be applied to more examples of EIS. To achieve better results, they should be applied by unbiased developers and reviewed by domain experts. By applying the results to more examples, and building a database of the best practices, the process model will mature.

- Several evaluation approaches are presented in the thesis. However, none of the software architecture evaluation approaches have been analysed in practice for the thesis. A future work is to analyse the current evaluation solutions for analysing the results of a goal model.
- EPF is used to develop tools to support the process models. The tools currently are published as static websites. To improve guidance and support for KAOS- β and EAPM, dynamic, web based tools are needed.
- EPF based tools also could interact with other tools specially the ones that could analyse and assess the process model and the results. Interacting with other tools could help to expand the domain of the tool.

9.4 Conclusion

A complex system is different to a complicated system. A computer is a complicated system, however, a complex system is a system for which the behaviour cannot be predicted by analysing components' interactions. Thus, considering the effect of human decisions, market changes, and business process concept, an EIS is a complex system (Wegmann, 2003). The way of thinking, the approaches to address the challenges, and the contributions are different for complex systems compare to complicated system.

To address the problem of EIS development, the development team (including domain experts) needs to make a paradigm shift from the mechanistic paradigm used to understand complicated systems to a systematic paradigm used to understand complex systems. Making explicit the existence of a systematic paradigm is an important contribution because it provides theoretical justification for what are the essential elements (here goals) of an EIS (Wegmann, 2003).

Paradigm is “a set of values, or principals that we use when we think” (Kuhn, 1962) as cited in (Wegmann, 2003). This thesis makes the existence of a systematic paradigm explicit by developing a method including process models to capture the values and goals of an EIS and use the information for an architecture design of an EIS.

The Implementation phase of EIS development, while important, is not the critical part of EIS development: the early phases – specifically those that are the focus of the thesis – are where essential complexity arise.

Appendix A

Results of Piloting ACM

Chapter 7 presents an example of EIS, Airport Crisis Management (ACM). ACM is used to pilot both KAOS- β and EAPM process models. The aim is to analyse these process models using a novel example of EIS. In addition, to demonstrate the relationships between these two process models and how the results of KAOS- β could be traced to the design phase. Figure 1.3 presents the roadmap that is followed in Chapter 7.

However, a sample result including the tables is presented in Chapter 7 and the further results are presented in this appendix. This helps Chapter 7 to be more focused on demonstrating how the steps are applied and what are a sample of expected results. This appendix is categorised in three sections. First, Section A.1 presents the further results of the fourth step of KAOS- β , which is documenting the goals. Section A.2 presents the further results of KAOS- β step six, which is documenting the links between the goals. Section A.3 presents the further results of final documentation of EAPM. The empty field of the tables could be filled by further analysis of the domain's data and brainstorming with a group of IT and domain specialists.

A.1 Step 4: Document goals

According to KAOS- β , after identifying modules and early goals, the next step is to document the goals. For documenting the goals, a template is suggested. This template is presented in Table A.1. Context in these tables are empty because no keyword found that require further explanation. However, to keep the look of the template constant the empty fields are not removed.

ID	ACMG4
Name	Effective Dissemination between Departments
Def	Effective dissemination to evaluate the situation, analyse the reports, and provide the solutions.
Scenario	Heads of rescue workers teams carry and disseminate using a PDA that is automatically configured in line with the type of incident and the role they have to play. They communicate all relevant information in real-time with a distributed blackboard application with other departments.
Context	
Priority	High
Source	ACM description and sample scenarios ¹
Issue/Notes	measure of success is related to the real-time communication

Table A.1: Structured documentation for ACM's goal with ID:ACMG4

ID	ACMG5
Name	Identify and Register Victims
Def	Identify and register victims as soon as possible with no errors.
Scenario	The list of passengers and crew names passed to the crisis management system for further investigation of the victims' situation.
Context	
Priority	Medium
Source	ACM description and sample scenarios ¹
Issue/Notes	measure of success: reliable and fast investigation of victims' situation

Table A.2: Structured documentation for ACM's goal with ID:ACMG5

ID	ACMG6
Name	Exchange Situational Information
Def	Exchange Situational Information between departments and rescue units
Scenario	The crisis management system provide integrated advance GIS system which allows visualising (at different levels of detail and with the appropriate properties) for different departments.
Context	
Priority	High
Source	ACM description and sample scenarios ¹
Issue/Notes	measure of success is the accuracy and speed of information, plus user friendly system and method of exchanging information.

Table A.3: Structured documentation for ACM's goal with ID:ACMG6

ID	ACMG7
Name	Communication Interface between Departments
Def	Different methods to help departments communicate with each other.
Scenario	The operation centre monitors the evacuation of the victims via a GIS system and interacts with hospital, police and rescue workers for a smooth evacuation of victims.
Context	
Priority	High
Source	ACM description and sample scenarios ¹
Issue/Notes	

Table A.4: Structured documentation for ACM's goal with ID:ACMG7

ID	ACMG8
Name	Central recording of victims
Def	Central recording of disaster victims and their appearance at a treatment facility.
Scenario	The list of registered plane crew and passengers would be compared against victims and will be passed to the treatment facilities.
Context	
Priority	Medium
Source	ACM description and sample scenarios ¹
Issue/Notes	measure of success: correct list and tracing the victims to the correct treatment facility within specific period of time.

Table A.5: Structured documentation for ACM's goal with ID:ACMG8

ID	ACMG9
Name	Pass Situational Information to relatives of victims
Def	Central organisation to pass real-time information to relatives.
Scenario	Meanwhile rescue process, correct information is passed to the help desk so that information could be provided to the relatives and other parties such as press etc.
Context	
Priority	Medium
Source	ACM description and sample scenarios ¹
Issue/Notes	Measure of success is related to the speed and accuracy of the information. This goal is the link to the Social Module .

Table A.6: Structured documentation for ACM's goal with ID:ACMG9

A.2 Step 6: Document links

According to KAOS- β after identifying modules, goals, and documenting the goals, the next step is to refine and document the links between the goals. To document the links, Table A.7 is suggested as a template.

ID	ACML1-4
Name	Link from ACMG1 to ACMG4
SysRef	Sys-to-be
Status	Goal under further refinement
Tactic	Designer assumption based on document ¹

Table A.7: Documentation for the link between ACMG1 and ACMG4

ID	ACML1-5
Name	Link from ACMG1 to ACMG5
SysRef	Sys-to-be
Status	Goal under further refinement
Tactic	Designer assumption based on document ¹

Table A.8: Documentation for the link between ACMG1 and ACMG5

ID	ACML2-6
Name	Link from ACMG2 to ACMG6
SysRef	Sys-to-be
Status	Goal under further refinement
Tactic	Designer assumption based on document ¹

Table A.9: Documentation for the link between ACMG2 and ACMG6

ID	ACML2-7
Name	Link from ACMG2 to ACMG7
SysRef	Sys-to-be
Status	Goal under further refinement
Tactic	Designer assumption based on document ¹

Table A.10: Documentation for the link between ACMG2 and ACMG7

ID	ACML5-8
Name	Link from ACMG5 to ACMG8
SysRef	Sys-to-be
Status	Goal under further refinement
Tactic	Designer assumption based on document ¹

Table A.11: Documentation for the link between ACMG5 and ACMG8

ID	ACML5-9
Name	Link from ACMG5 to ACMG9
SysRef	Sys-to-be
Status	Goal under further refinement
Tactic	Designer assumption based on document ¹

Table A.12: Documentation for the link between ACMG5 and ACMG9

A.3 EAPM Tables

According to EAPM, while applying all the steps, to document the results a template, presented as table A.13, is used. The empty fields in these tables could be filled by further analysis of the domain's data and brainstorming with IT and domain specialists.

Attribute (s)	Safe DB
Scenario (s)	The information is saved and retrieved to and from database under safety regulations
Goal (s)	ACMG8, ACMG4
Environment	Database (s) operates under defined safety standards
Stimulus	Insert or retrieved information to and from database (s)
Response	Check the inserted data and retrieved requests against security criteria
Measure	No unreliable source can insert or retrieve information from database (s)
Strategy	Database masking and wallet encryption ¹
Reasoning	Wallet encryption prevents from theft and masking prevents from everyday misuse of data
Architectural diagram	

Table A.13: Architectural information for Safe Database quality attribute.

Attribute (s)	Different Communication Devices
Scenario (s)	The rescue workers are providing first aid to the victims and are recording all data on their PDA's. The first victim is ready to be transported to the hospital. The operation centre communicates the correct location of the hospital to the rescue workers.
Goal (s)	ACMG11, ACMG12
Environment	The system accepts information and transactions from and to different devices and send information to different devices.
Stimulus	Communication with external devices
Response	System communicates with different defined devices
Measure	No failure response from different devices
Strategy	Different hardware and software interfaces
Reasoning	As much as technology allows
Architectural diagram	

Table A.14: Architectural information for Different Communication Devices quality attribute.

Attribute (s)	Supporting Different Interface
Scenario (s)	The system offers the integration of advanced GIS systems which allows visualising (at different levels of detail and with the appropriate properties) the position of any number of victims and resources.
Goal (s)	ACMG11, ACMG12, ACMG18, ACMG19, ACMG20
Environment	The system accepts information and transactions from and to different devices and send information to different devices.
Stimulus	Communication with external devices
Response	System communicates with different interfaces (Human and Machine)
Measure	No failure response from different interfaces
Strategy	Web-base interfaces
Reasoning	Helps to design different, fast, and cost effective interfaces for some of the hardware
Architectural diagram	

Table A.15: Architectural information for Supporting Different Interface quality attribute.

Attribute (s)	Availability
Scenario (s)	The fire department fail to receive the live video of the fire incident, hence a technical witness sends the information via phone.
Goal (s)	ACMG17
Environment	The system accepts information and transactions from alternative sources.
Stimulus	Communication with alternative sources and devices.
Response	System communicates with different sources.
Measure	No failure response in functionality of the system
Strategy	Run the system and database across a cluster of servers. Provide alternative human and non-human sources.
Reasoning	Provides 24/7 availability, on-demand scalability ²
Architectural diagram	

Table A.16: Architectural information for **Availability** quality attribute.

Attribute (s)	Modifiability
Scenario (s)	The system is designed in such that supports the future changes; for example, in the future the system should be connected directly to the government security department.
Goal (s)	ACMG17, ACMG8
Environment	The system integrate with new systems or sub-systems.
Stimulus	New components for communicating with the security department.
Response	Developers add the new components.
Measure	The system communicates with the security department via the new components.
Strategy	Localise changes (anticipate expected changes, generalize module, and limit possible options)
Reasoning	Provide suitable access to the right level of EIS data and functionalities.
Architectural diagram	

Table A.17: Architectural information for **Modifiability** quality attribute.

Attribute (s)	Performance
Scenario (s)	An aircraft is crashed in a rural area. The system should collect information about the number of victims in less than an hour.
Goal (s)	ACMG5
Environment	The system starts working with the information about the accident and victims.
Stimulus	Receiving information about an accident.
Response	System collect, store, analyse, and distribute the latest information about the accident.
Measure	The system collect data about the victims in less than an hour.
Strategy	Oracle real application clusters and make performance model.
Reasoning	Runs faster than the fastest mainframe (Oracle, 2011b).
Architectural diagram	

Table A.18: Architectural information for Performance quality attribute.

Attribute (s)	Cost-Benefit and time to market
Scenario (s)	The system should be able to expand in short time and within allocated budget.
Goal (s)	goals from business module
Environment	Normal environment
Stimulus	Request for expanding the system.
Response	The system is expanded within budget by adding new servers.
Measure	The system adds new servers to support the expansion of the system.
Strategy	Oracle real application clusters
Reasoning	Expand capacity by simply adding servers to your cluster (Oracle, 2011b).
Architectural diagram	

Table A.19: Architectural information for Cost-Benefit and time to market quality attribute.

Glossary

A

Agents An agent is an active system component playing a specific role in goal satisfaction (van Lamsweerde, 2009, p. 260).

Architectural Strategy An overall plan rather than detailed tactic to accomplish an architecture.

Architectural Style Expresses a fundamental structural organization scheme for software systems. It provides a set of predefined element types, specific their responsibilities, and includes rules and guidelines for organizing the relationship between them (Rozanski & Woods, 2005).

B

Business Processes A set of partially ordered steps intended to reach a goal (Curtis et al., 1992, p. 76).

E

EIS Architecture Process Model (EAPM) A process model to model a transition from the goal structure to starting point of EIS architecture design.

Enterprise Information Systems (EIS) An EIS is a software system that integrates the business processes of organisation(s) to improve their functioning (Tabatabaie et al., 2008).

G

Goal Goals are high level objectives of the business, organization, or system. They express the rationale for proposed systems and

guide decisions at various levels within the enterprise (Antón, 1996, p.137).

K

KAOS- β A process model for explicitly eliciting, capturing, and specifying the goals of an EIS.

P

Process Model Nested set of abstractions intended to reach a goal (Feiler & Humphrey, 1993).

R

Requirements Engineering (RE) The process of finding out, analysing, documenting and checking these services and constraints is called requirement engineering (Sommerville, 2007, p. 118).

S

Scenarios Scenarios are behavioural descriptions of a system and its environment arising from restricted situations. They exemplify behaviours enabling hidden needs to be uncovered and are useful for evaluating design alternatives and validating designs (Antón, 1996, p. 138).

Service Oriented Architecture (SOA) SOA is one of the most effective tools to use for incremental integration of legacy application functionality. Once an interface (contract) has been defined for some part of a legacy application's services, plugging that interface into an SOA is relatively straightforward (McGovern et al., 2003, p. 25).

Software Architecture The software architecture of a program or computing system is the structure or structures of the system, which comprise software elements, the externally visible properties of those elements, and the relationships among them (Bass et al., 2003).

Bibliography

- ACM (2010). Aviation Crisis Management 2010. [Accessed February 2011]
Available at: <http://www.internationalairportreview.com/>.
- Acute Trusts (2011). Authorities and trusts. [Accessed February 2011]
Available at: <http://www.nhs.uk/NHSEngland/thenhs/about/Pages/authoritiesandtrusts.aspx>.
- Aguilar, E. R., Ruiz, F., García, F., & Piattini, M. (2006). Applying software metrics to evaluate business process models. *CLEI Electron. J.*, 9(1–15).
- Alegría, J. A. H. (2011). Analyzing Software Process Models with AVISPA. In *International Conference on Software and Systems Process*.
- An, Y., Dalrymple, P., Rogers, M., Gerrity, P., Horkoff, J., & Yu, E. (2009). Collaborative social modeling for designing a patient wellness tracking system in a nurse-managed healthcare center. In *International Conference on Design Science Research in Information Systems and Technology*, pp. 1–14. ACM.
- Antón, A. I. (1996). Goal-based requirements analysis. In *IEEE International Conference on Requirements Engineering*, pp. 136–144. IEEE Computer Society.
- Anyanwu, K., Sheth, A., Cardoso, J., Miller, J., & Kochut, K. (2003). Healthcare enterprise process development and integration. *Journal of Research and Practice in Information Technology, Special Issue in Health Knowledge Management*, 35(2), 83–98.
- Armour, F., Kaisler, S., & Liu, S. (1999). A big-picture look at enterprise architectures. *IT Professional*, 1(1), 35–42.
- Bass, L., Clements, P., & Kazman, R. (2003). *Software Architecture in Practice* (2nd edition). Addison-Wesley Professional.

- Bassry and Associates (2010). Design Decomposition for Business Process and Data Flow Diagrams. [Accessed December 2010] Available at: <http://www.designdecomposition.com/>.
- Berg, M. (2001). Implementing information systems in health care organizations: myths and challenges. *International Journal of Medical Informatics*, 64(2), 143–156.
- Birman, K. P., & Joseph, T. A. (1987). Reliable communication in the presence of failures. *ACM Transactions on Computer Systems*, 5(1), 47–78.
- Blum, B. I. (1994). A taxonomy of software development methods. *Journal of Enterprise Architecture*, 37(11), 82–94.
- Bowers, S. (2010). Where the NHS's software scheme went wrong. [Accessed March 2011] Available at: <http://www.guardian.co.uk/business/2010/mar/21/nhs-national-program-problems>.
- Bull (1998). Failure causes statistics. [Accessed February 2011] Available at: http://www.it-cortex.com/Stat_Failure_Cause.htm.
- Castro, J., Kolp, M., & Mylopoulos, J. (2002). Towards Requirements-Driven Information Systems Engineering: The Tropos Project. *Information Systems*, 27(6), 365–389.
- Chroust, G., Kuhrmann, M., & Schoitsch, E. (2010). Modeling software development processes. In Cruz-Cunha, M. M. (Ed.), *Social, Managerial, and Organisational dimensions of enterprise information systems*, pp. 31–57. IGI.
- Clements, P., & Bass, L. (2010a). Business goals as architectural knowledge. In *Proceedings of the 2010 ICSE Workshop on Sharing and Reusing Architectural Knowledge*, pp. 9–12. ACM.
- Clements, P., & Bass, L. (2010b). Relating business goals to architecturally significant requirements for software systems. Tech. rep. CMU/SEI-2010-TN-018, Carnegie Mellon.
- Clements, P., Kazman, R., & Klein, M. (2002). *Evaluating Software Architectures: Methods and Case Studies*. Addison-Wesley.

- Collins, T. (2006). Major incidents hit NHS national systems. [Accessed March 2011] Available at: <http://www.computerweekly.com/Articles/2006/09/19/218552/Major-incidents-hit-NHS-national-systems.htm>.
- Critical Incident Response Group (2007). Critical Information TechnologyORION. [Accessed September 2011] Available at: <http://www.fbi.gov/about-us/cirg/investigations-and-operations-support>.
- Curtis, B., Kellner, M. I., & Over, J. (1992). Process modeling. *Communications of the ACM*, 35(9), 75–90.
- Darimont, R., Delor, E., Massonet, P., & van Lamsweerde, A. (1997). GRAIL/KAOS: An Environment for Goal-Driven Requirements Engineering. In *IEEE International Symposium on Requirements Engineering*, pp. 612–613. IEEE Computer Society.
- Data corruption (2011). Data corruption. [Accessed February 2011] Available at: http://en.wikipedia.org/wiki/Data_corruption.
- Delor, E., Darimont, R., & Rifaut, A. (2009). Software Quality Starts with the Modelling of Goal-Oriented Requirements. [Accessed December 2010] Available at: <http://www.objectiver.com>.
- DH Stroke Policy (2007). National Stroke Strategy. [Accessed March 2010] Available at: <http://www.dh.gov.uk/en/Publicationsandstatistics/Publications/>.
- Dictionary.com (2011). Philosophy—Dictionary.com. [Accessed August 2011] Available at: <http://dictionary.reference.com/browse/philosophy>.
- DODAF (2007). DoD Architecture Framework Version 1.5. Tech. rep., Department of Defence.
- DODAF (2009). DoD Architecture Framework Version 2.0. Tech. rep., Department of Defence.
- Dumas, M., & ter Hofstede, A. H. (2001). UML activity diagrams as a workflow specification language. In *International Conference on The Unified Modeling Language, Modeling Languages, Concepts, and Tools*, Vol. 2185 of *LNCS*, pp. 76–90. Springer.

- ECRI Institute (2011). Top 10 health technology hazards for 2011. [Accessed November 2011] Available at: https://www.ecri.org/Documents/Secure/Health_Devices_Top_10_Hazards_2012.pdf.
- Edward, C., Ward, J., & Bytheway, A. (1993). *The Essence of Information Systems* (2nd edition). Prentice Hall.
- Electric, G. (2011). Product and services. [Accessed February 2011] Available at: http://www.ge.com/products_services/index.html.
- EmerGeo (2011). EmerGeo emergency and crisis management software. [Accessed September 2011] Available at: <http://www.emergeo.com/>.
- EPF Project (2009). Eclipse Process Framework Project (EPF). [Accessed May 2009] Available at: <http://www.eclipse.org/epf/>.
- EPF Project (2010). Introduction to the Eclipse Process Framework. [Accessed 30 March 2010] Available at: <http://epf.eclipse.org>.
- Erl, T. (2010). *Service-Oriented Architecture: Concepts, Technology, and Design*. Prentice Hall.
- ERMS (2011). Erms crisis manager - embedded crisis management software. [Accessed September 2011] Available at: <http://www.ermscorp.com/crisis-management-software/>.
- Feiler, P. H., & Humphrey, W. S. (1993). Software process development and enactment: Concepts and definitions. In *Second International Conference on the Software Process: Continuous Software Process Improvement*, pp. 28–40. IEEE Computer Society.
- Finkelstein, A. (1993). Report of Inquiry Into The London Ambulance Service. Tech. rep. 0905133706, University College London.
- Fruin, M. (1994). *The Japanese Enterprise System: Competitive Strategies and Cooperative Structures*. Oxford University Press.
- Fuggetta, A. (2000). Software process: a roadmap. In *ICSE - Future of SE Track*, pp. 25–34.
- Garcia, F., Piattini, M., Ruiz, F., Canfora, G., & Visaggio, C. A. (2006). FMESP: Framework for the modeling and evaluation of software processes. *Journal of Systems Architecture*, 52, 627–639.

- Garlan, D., & Shaw, M. (1994). An Introduction To Software Architecture. In Ambriola, V., & Tortora, G. (Eds.), *Advances in Software Engineering and Knowledge Engineering (Series in Software Engineering & Knowledge Engineering)* (8th edition), pp. 1–40. World Scientific Publishing Co Pte Ltd.
- German Telekom (2011). A quick guide to key technical terms. [Accessed March 2011] Available at: <http://www.telekominnovationcenter.de/dtag>.
- Gopalkrishnan, J., & Gupta, V. (2007). ebay: “the worlds largest online marketplace” - a case study. In *Global Competition and Competitiveness of Indian Corporate*, pp. 543–549.
- Gorelic, P. B. (2009). Challenges of designing trials for the primary prevention of stroke. *American Stroke Association*, 40, 82–84.
- Gruhn, V. (1991). Validation and verification of software process models. In *European symposium on Software Development Environments and CASE Technology*, pp. 271–286. Springer.
- Habli, I., Wu, W., Attwood, K., & Kelly, T. (2007). Extending Argumentation to Goal-Oriented Requirements Engineering . In *Advances in Conceptual Modeling - Foundations and Applications: ER Workshops*, Vol. 4802 of *LNCS*, pp. 306–316. Springer.
- Harrell, E. (2009). In Denmark’s Electronic Health Records Program, a Lesson for the U.S. [Accessed September 2009] Available at: <http://www.time.com>.
- Haumer, P. (2011). Eclipse Process Framework Composer, Part 1: Key Concepts. [Accessed February 2011] Available at: <http://www.eclipse.org/epf/general/EPFComposerOverviewPart1.pdf>.
- Hong, J. (2001). Goal recognition through goal graph analysis. *Journal of Artificial Intelligence Research*, 15, 1–30.
- Hoogendoorn, M., Jonker, C., Maanen, P., & Treur, J. (2009). Agent-Based Analysis and Simulation of Meta-Reasoning Processes in Strategic Naval Planning. *Knowledge-Based Systems Journal*, 22, 589–599.
- Humphrey, W. S. (2007). Software process improvement A personal view: How it started and where it is going: Research Sections. *Software Process: Improvement and Practice*, 12(3), 223–227.

- Humphrey, W. S., & Kellner, M. I. (1989). Software process modeling: principles of entity process models. In *Proceedings of the 11th International Conference on Software Engineering*, pp. 331–342. ACM.
- Iannella, R., Robinson, K., & Rinta-Koski, O. (2007). Towards A Framework For Crisis Information Management Systems (CIMS). [Accessed September 2011] Available at: <http://cairns.sourceforge.net/tiems2007.pdf>.
- IBM (2005). The National Danish e-health Portal. [Accessed May 2009] Available at: <http://www-05.ibm.com/services/dk/gbs/healthcare/>.
- IBM Software Information Center (2007). TXSeries for Multiplatforms Version 6.2. [Accessed June 2011] Available at: <http://publib.boulder.ibm.com/infocenter/txformp/v6r2/index.jsp?topic=/com.ibm.cics.tx.doc/index.htm>.
- International Airport Review (2009). Aviation Crisis Management 2009. [Accessed August 2011] Available at: <http://www.internationalairportreview.com/1025/events/aviation-crisis-management-2009/>.
- International Airport Review (2011). Airport crisis management - Articles and news items. [Accessed May 2011] Available at: <http://www.internationalairportreview.com/tag/airport-crisis-management/>.
- Jordans, F., & Lekic, S. (2010). Volcanic ash shuts down European airports. [Accessed February 2011] Available at: <http://www.msnbc.msn.com/id/37041388/ns/travel-news/>.
- Kaplan, B., & Harris-Salamone, K. D. (2009). Health it success and failure: Recommendations from literature and an amia workshop. *Journal of the American Medical Informatics Association*, 16(3), 291–299.
- Kavakli, E., & Loucopoulos, P. (2005). Goal modeling in requirements engineering: Analysis and critique of current methods. In Krogstie, J., Halpin, T. A., & Siau, K. (Eds.), *Information modeling methods and methodologies*, pp. 102–124. IGI.
- Kavakli, E., Loucopoulos, P., & Filippidou, D. (1996). Using scenarios to systematically support goal-directed elaboration for information system requirements. In *IEEE Symposium and Workshop on Engineering of Computer Based Systems*, pp. 308–314. IEEE Computer Society.

- Keef, M. (2011). Top 10 corporate information technology failures. [Accessed February 2011] Available at: <http://www.computerworld.com/computerworld/records/images/pdf/44NfailChart.pdf>.
- Kelly, T. (1998a). A six-step Method for Developing Arguments in the Goal Structuring Notation (GSN). [Accessed November 2011] Available at: <http://www.origin-consulting.com/gsnclub/gsnmethod.pdf>.
- Kelly, T. (1998b). *Arguing Safety – A systematic approach to managing Safety Cases*. Ph.D. thesis, University of York, Department of Computer Science.
- Kelly, T. (2001). Concepts and principles of compositional safety case construction. Tech. rep. COMSA/2001/1/1, University of York.
- Kelly, T. (2004). A Systematic Approach to Safety Case Management. In *SAE 2004 World Congress*. Society for Automotive Engineers.
- Kelly, T., & Weaver, R. (2004). The Goal Structuring Notation: A Safety Argument Notation. In *Proceedings of the Dependable Systems and Networks 2004 Workshop on Assurance Cases*. IEEE Computer Society.
- Kim, J. S., Park, S., & Sugumaran, V. (2006). Contextual problem detection and management during software execution in complex environments. *Industrial Management and Data Systems*, 106, 540–561.
- Kuhn, T. S. (1962). *The Structure of Scientific Revolutions*. University of Chicago Press.
- Lankhorst, M. (2005). *Enterprise Architecture at Work: Modelling, Communication and Analysis*. Springer.
- Laudon, J. P., & Laudon, K. C. (2007). *Management Information Systems: Managing the Digital Firm* (10th edition). Prentice Hall.
- Lazoff, D., & Stephens, A. (1996). Optimal-availability placement of replicated data in distributed systems. In *Proceedings of the 1996 IEEE Fifteenth Annual International Phoenix Conference on Computers and Communications*. IEEE Computer Society.
- Levine, S. R., & Gorman, M. (1999). Telectroke: The Application of Telemedicine for Stroke. *American Stroke Association*, 1, 464–469.

- Liu, K., Fox, M., Apers, P., Klein, M., Cheng, A., Stamper, R., Chattopadhyay, S., & Greene, T. (2000). Enterprise information systems: issues, challenges and viewpoints. In Filipe, J. (Ed.), *Enterprise information systems*, pp. 1–13. Kluwer Academic Publishers.
- Ma, X., Vazhkudai, S., & Zhang, Z. (2009). Improving Data Availability for Better Access Performance: A Study on Caching Scientific Data on Distributed Desktop Workstations. *Journal of Grid Computing*, 7, 419–438.
- Magister Ludia Aviation and Softsolutions (2011). Aviation Crisis Management 2009. [Accessed August 2011] Available at: http://www.magisterludi.com/public_html/aviation/consultancy/pdf/APTSpresntn.pdf.
- McGovern, J., Ambler, S., Stevens, M., Linn, J., Sharan, V., & Jo, E. (2003). *Practical Guide to Enterprise Architecture*. Prentice Hall.
- Meyer, B. (2000). *Object-Oriented Software Construction* (2nd edition). Prentice Hall.
- Minoli, D. (2008). *Enterprise Architecture A to Z: Frameworks, Business Process Modeling, SOA, and Infrastructure Technology*. Taylor & Francis.
- Mitra, T. (2008). Part 6: Why business process management (BPM) is important to an enterprise. [Accessed February 2011] Available at: <http://www.ibm.com/developerworks/webservices/library/ar-arprac6/index.html>.
- Mitsubishi Committee (2011). About mitsubishi. [Accessed February 2011] Available at: <http://www.mitsubishi.com/e/group/about.html>.
- Mitsubishi Logistics (2011). Information System Services. [Accessed May 2011] Available at: <http://www.mitsubishi-logistics.co.jp/english/service/pd/system/index.html>.
- MODAF (2010). Documentation supporting the MOD Architecture Framework (MODAF). [Accessed September 2010] Available at: <http://www.mod.uk/DefenceInternet/AboutDefence/WhatWeDo/InformationManagement/MODAF/ModafDetailedGuidance.htm>.
- MODELPLEX Consortium (2007). Deliverable D1.1a: Case Study Scenario Definitions. [Accessed May 2011] Available at: <http://www.modelplex.org/>.

- Molinaro, L. F. R., Ramos, K. H. C., da Cotta Orlandi, T. R., & Abdalla, H. (2010). Enterprise Architecture to IT Governance: An Approach Based on Component Business Model and Performance Levels. In *Communications in Computer and Information Science*, pp. 41–51. Springer.
- Morse, A. (2011). The National Programme for IT in the NHS: an update on the delivery of detailed care records systems. Tech. rep. HC888, National Audit Office.
- Murata, T. (1989). Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4), 541–580.
- Nagy, Z., Simon, P., Sipos, E., & Kozmann, G. (1995). The main elements of the information system of the National Stroke Program (Smart Card - Telecommunication - Knowledge Bases). [Accessed September 2011] Available at: <http://www.ncbi.nlm.nih.gov/pubmed>.
- NATCA (2011). How We Guide You Home. [Accessed August 2011] Available at: <http://www.natca.org/>.
- National Institute for Health and Clinical Excellence (2008). Information about NICE clinical guideline. [Accessed August 2011] Available at: <http://www.nice.org.uk/nicemedia/live/12018/41315/41315.pdf>.
- Nedstam, J., & Staples, M. (2007). Evolving strategies for software architecture and reuse. *Software Process: Improvement and Practice*, 12, 295–309.
- Neighbors, J. M. (1980). Software construction using components. [Accessed May 2011] Available at: <http://www.bayfronttechnologies.com/thesis.htm>.
- Neubauer, M. J. (2007). A systems analysis of information technology and the use of WLANs Implemented by an FBO Field Office for Crisis Response Incidents: The Columbia Field Office Case Study. [Accessed September 2011] Available at: <http://books.google.co.uk/books?id=1kRp0j3qc6wC&printsec=frontcover#v=onepage&q&f=false>.
- NHS (2011). Treating Stroke. [Accessed August 2011] Available at: <http://www.nhs.uk/Conditions/Stroke/Pages/treatment.aspx>.
- Noland, D. (2011). 10 Plane Crashes That Changed Aviation. [Accessed February 2011] Available at: <http://www.popularmechanics.com/technology/aviation/crashes/>.

- Nuseibeh, B., & Easterbrook, S. (2000). Requirements engineering: A roadmap. In *International Conference on Software Engineering*, pp. 35–46. ACM Press.
- Object Management Group (2008). Software & Systems Process Engineering Meta-Model Specification. [Accessed January 2011] Available at: <http://www.omg.org/spec/SPEM>.
- Objectiver (2010). Objectiver Tool. [Accessed February 2011] Available at: <http://www.objectiver.com/>.
- O'Brien, L., Bass, L., & Merson, P. (2005). Quality attributes and service-oriented architectures. Tech. rep. CMU/SEI-2005-TN-014, Software Engineering Institute, Carnegie Mellon.
- Oracle (2011a). Oracle Loyalty Analytics. [Accessed September 2011] Available at: <http://www.oracle.com/us/solutions/ent-performance-bi/loyalty-analytics-066542.html>.
- Oracle (2011b). Oracle Real Application Clusters. [Accessed May 2011] Available at: <http://www.oracle.com/us/products/database/options/real-application-clusters/index.html>.
- Páscoa, C., Belo, N., & José (2010). Value model for enterprise and process architecture alignment verification. In *Communications in Computer and Information Science*, pp. 63–72. Springer.
- Peralta, V. (2006). Data Freshness and Data Accuracy: A State of the Art. [Accessed June 2011] Available at: <http://www.fing.edu.uy/inco/pedeciba/bibliote/reptec/TR0613.pdf>.
- Pereira, E. B., Bastos, R. M., da C. Móra, M., & Oliveira, T. C. (2011). IMPROVING THE CONSISTENCY OF SPEM-BASED SOFTWARE PROCESSES. In *13th International Conference on Enterprise Information Systems*, pp. 1–10. Science and Technology Publications.
- Perrig, A., Szewczyk, R., Wen, V., Culler, D., & Tygar, J. D. (2001). SPINS: Security Protocols for Sensor Networks. In *The Annual International Conference on Mobile Computing and Networking*, pp. 189–199. ACM.
- Perry, D. E., & Wolf, A. L. (1992). Foundations for the study of software architecture. *SIGSOFT Software Engineering Notes*, 17(4), 40–52.

- Pradhan, A., Laefer, D. F., & Rasdorf, W. J. (2007). Infrastructure management information system framework requirements for disasters. *Computing in Civil Engineering*, 21(2), 90–101.
- Public Health Data Standards Consortium (2011). Privacy and Security Standards. [Accessed June 2011] Available at: http://www.phdsc.org/standards/health-information/PS_Standards.asp.
- RAE-BCS Working Group (2004). The Challenges of Complex IT Projects. Tech. rep. 1-903496-15-2, The Royal Academy of Engineering.
- Ramsin, R. (2006). *The Engineering of an Object-Oriented Software Development Methodology*. Ph.D. thesis, University of York, Department of Computer Science.
- Regev, G., & Wegmann, A. (2005). Where do goals come from: the underlying principles of goal-oriented requirements engineering. In *International Conference on Requirements Engineering*, pp. 253–362. IEEE Computer Society.
- Rinkineva, K. (2004). The role of information technology in crisis management. [Accessed September 2011] Available at: http://www.einiras.org/conf/conferences/documents/Information_Technology_in_Crisis_ManagementEINIRAS.pdf.
- Romano, M. J., & Stafford, R. S. (2011). Electronic health records and clinical decision support systems. *Archives of Internal Medicine*, 171(10), 897–903.
- Royal Academy Engineering (2004). The challenges of complex IT projects. Tech. rep., The Royal Academy of Engineering.
- Rozanski, N., & Woods, E. (2005). *Software Systems Architecture: Working With Stakeholders Using Viewpoints and Perspectives*. Addison-Wesley.
- Schwamm, L., Audebert, H. J., Amarenco, P., Chumbler, N. R., Frankel, M. R., George, M. G., Gorelick, P. B., Horton, K. B., Kaste, M., Lackland, D. T., Levine, S. R., Meyer, B. C., Meyers, P. M., Patterson, V., Stranne, S. K., & White, C. J. (2009). Recommendations for the Implementation of Telemedicine Within Stroke Systems of Care, A Policy Statement From the American Heart Association. *American Stroke Association*, 1, 1–26.

- Schwamm, L., Pancioli, A., Acker, J., Goldstein, L., Zorowitz, R., Shephard, T., Moyer, P., Gorman, M., Johnston, C., Duncan, P., Gorelick, P., Frank, J., Stranne, S., Smith, R., Federspiel, W., Horton, K., Magnis, E., & Adams, R. (2005). Recommendations for the establishment of stroke systems of care. *American Stroke Association*, 36, 690–703.
- Shanmuganathan, S. (2010). A Stroke Information System (SIS): Critical Issues and Solutions. In Pease, W., Cooper, M., & Gururajan, R. (Eds.), *Biomedical Knowledge Management: Infrastructures and Processes for E-Health Systems*, pp. 177–191. IGI.
- Soffer, P., & Wand, Y. (2004). Goal-driven analysis of process model validity. In *Proceedings of Advanced Information Systems Engineering*, pp. 521–535. Springer-Verlag.
- Sommerville, I. (2007). *Software Engineering* (8th edition). Addison Wesley.
- Song, X. (1995). A framework for understanding the integration of design methodologies. *ACM SIGSOFT Software Engineering Notes*, 20(1), 46–54.
- Soumerai, S., & Avery, T. (2010). Don't Repeat the UK's Electronic Health Records Failure. [Accessed September 2011] Available at: http://www.huffingtonpost.com/stephen-soumerai/dont-repeat-the-uks-elect_b_790470.html.
- SPEM (2009). Software Process Engineering Meta-Model (SPEM). [Accessed May 2009] Available at: www.omg.org/technology/documents/formal/spem.htm.
- Stair, R. M., & Reynolds, G. (2006). *Principles of Information Systems* (7th edition). Thomson.
- Stevens, J., Heckendorn, R. B., & Soule, T. (2005). Exploiting disruption aversion to control code bloat. In *Generic and Evolutionary Computation Conference*, pp. 1605–16012. ACM.
- Stevens, P., & Pooley, R. (1999). *Using UML : Software Engineering With Objects and Components*. Addison Wesley.
- strokeprevention.org (2011). Stroke Prevention. [Accessed August 2011] Available at: <http://www.strokeprevention.org/>.

- Tabatabaie, M. (2009). Applying GSN to Stroke Care. [Accessed May 2011] Available at: www-users.cs.york.ac.uk/malihetb/Publication/GRR-Main.pdf.
- Tabatabaie, M. (2010). Implementing a tool to Support KAOS- β Process Model Using EPF. [Accessed August 2011] Available at: <http://www-users.cs.york.ac.uk/malihetb/Publication/Eclipse-process-Framework-Step-by-step-example.pdf>.
- Tabatabaie, M. (2011). EIS Technologies and Terminologies. [Accessed August 2011] Available at: <http://www-users.cs.york.ac.uk/malihetb/Publication/EISTechReview.pdf>.
- Tabatabaie, M., Paige, R., & Kimble, C. (2010). Exploring enterprise information systems. In Cruz-Cunha, M. M. (Ed.), *Social, Managerial, and Organisational dimensions of enterprise information systems*, pp. 415–432. IGI.
- Tabatabaie, M., Paige, R. F., & Kimble, C. (2008). Exploring the boundaries of enterprise information systems. In *Second York Doctoral Symposium on Computing*, Vol. YCS-2008-434, pp. 27–34. Department of Computer Science, University of York.
- Tabatabaie, M., Polack, F. A., & Paige, R. (2010a). Evaluating goal-oriented analysis in the domain of enterprise information systems. In *Conference on Enterprise Information Systems (CENTERIS)*, pp. 62–71. Springer.
- Tabatabaie, M., Polack, F. A., & Paige, R. (2010b). KAOS- β : A Goal-Oriented Process Model for EIS. In *8th international workshop on modelling, simulation, verification and validation of enterprise information systems*, pp. 40–49. Science and Technology Publications.
- The Danish Evaluation Institute (2004). Criteria based evaluations, EVA's experience in evaluations based on criteria. [Accessed June 2011] Available at: <http://english.eva.dk/>.
- TheFreeDictionary (2011). Analytical. [Accessed June 2011] Available at: <http://www.thefreedictionary.com/analytical>.
- TOGAF (2009). Welcome to TOGAF Version 9 – The Open Group Architecture Framework. Tech. rep. 978-90-8753-230-7, The Open Group.
- Toulmin, S. (1958). *The Uses of Argument*. Cambridge University Press.

- van der Aalst, W. (2003). Don't go with the flow: Web services composition standards exposed. *IEEE Intelligent Systems*, 18, 72–76.
- van Lamsweerde, A. (2001). Goal-oriented requirements engineering: A guided tour. In *IEEE International Conference on Requirements Engineering*, pp. 249–262. IEEE Computer Society.
- van Lamsweerde, A. (2003). From system goals to software architecture. *Formal Methods for Software Architectures*, 2804, 25–43.
- van Lamsweerde, A. (2004). Goal-oriented requirements engineering: A roundtrip from research to practice. *IEEE International Conference on Requirements Engineering*, 4–7.
- van Lamsweerde, A. (2009). *Requirements Engineering: From System Goals to UML Models to Software Specifications*. Wiley.
- Weber-Jahnke, J., & Onabajo, A. (2009). Mining and analysing security goal models in health information systems. In *Workshop on Software Engineering in Health Care*, pp. 42–52. IEEE Computer Society.
- Wegmann, A. (2003). The Systemic Enterprise Architecture Methodology (SEAM) - Business and IT Alignment for Competitiveness. In *International Conference on Enterprise Information Systems*, pp. 483–490.
- Weiss, D. M., Bennett, D., Payseur, J. Y., Tendick, P., & Zhang, P. (2002). Goal-oriented software assessment. In *International Conference on Software Engineering*, pp. 221–231. ACM.
- White, S. A. (2006). Introduction to BPMN. [Accessed February 2011] Available at: http://www.bpmn.org/Documents/OMG_BPMN_Tutorial.pdf.
- Wikipedia (2011a). Amazon.com — Wikipedia, The Free Encyclopedia. [Accessed August 2011] Available at: <http://en.wikipedia.org/wiki/Amazon.com>.
- Wikipedia (2011b). Business Process Execution Language — Wikipedia, The Free Encyclopedia. [Accessed June 2011] Available at: http://en.wikipedia.org/wiki/Business_Process_Execution_Language.
- Wikipedia (2011c). Business Process Model and Notation — Wikipedia, The Free Encyclopedia. [Accessed June 2011] Available at: http://en.wikipedia.org/wiki/Business_Process_Modeling_Notation.

- Wikipedia (2011d). Enterprise Information System — Wikipedia, The Free Encyclopedia. [Accessed May 2011] Available at: http://en.wikipedia.org/wiki/Enterprise_Information_System.
- Wikipedia (2011e). Flowchart — Wikipedia, The Free Encyclopedia. [Accessed June 2011] Available at: http://en.wikipedia.org/wiki/Code_bloat.
- Wikipedia (2011f). Flowchart — Wikipedia, The Free Encyclopedia. [Accessed June 2011] Available at: <http://en.wikipedia.org/wiki/Flowchart>.
- Wikipedia (2011g). Two-factor authentication — Wikipedia, The Free Encyclopedia. [Accessed June 2011] Available at: http://en.wikipedia.org/wiki/Two-factor_authentication.
- Yu, E., & Mylopoulos, J. (1994). Towards modelling strategic actor relationships for information systems development – with examples from business process reengineering. In *Workshop on Information Technologies and Systems*, pp. 21–28.
- Yu, E. S. K., Mylopoulos, J., & Lespérance, Y. (1996). AI Models for Business Process Reengineering. *IEEE Expert: Intelligent Systems and Their Applications*, 11, 16–23.
- Zachman, J. A. (1987). A framework for information systems architecture. *IBM System Journal*, 26(3), 276–292.