

A Conceptual Framework for Simulating Feedback Loops in Engineering Design

Francisco Tapia Lara

Submitted in accordance with the requirements for the degree of
Doctor of Philosophy

The University of Leeds
School of Mechanical Engineering

February 2023

The candidate confirms that the work submitted is his/her own, except where work which has formed part of jointly authored publications has been included. The contribution of the candidate and the other authors to this work has been explicitly indicated below. The candidate confirms that appropriate credit has been given within the thesis where reference has been made to the work of others. *Further details of the jointly authored publications and the contributions of the candidate and the other authors to the work should be included below this statement.*

The work in Chapter 3, sections 3.2.2 and 3.2.3 and Chapter 5 section 5.1.2 Discrete events and 5.3.1 validation of the discrete events model include sections of the publication:

The thesis references the co-authored work in chapters, 4,5,6.

Simulation of Feedback Loops in Engineering Design. *Proceedings of the Design Society*. August 2021 Volume 1: ICED21, **1**, pp.2661-2670.

This copy has been supplied on the understanding that it is copyright material and that no quotation from the thesis may be published without proper acknowledgement.

Francisco Tapia was responsible for the development the discrete events simulation model and results of the simulation and literature review. The co-authors contributed with the addressing the adequate theoretical foundations and providing feedback and comments for the writing of the abstract, introduction, and conclusions.

This copy has been supplied on the understanding that is copyright material and that no quotation from the thesis can be published without proper acknowledgement.

The right of Francisco Tapia Lara to be identified as Author of this work has been asserted by him in accordance with the Copyright, Designs and Patents Act 1988.

Acknowledgements

I want to express my sincere gratitude to my supervisors, Professor Alison McKay, and Dr Mark Robinson, for their tireless support, encouragement, and guidance throughout the PhD. None of my work would have been possible without their kind supervision and teachings.

I would like to express my special thanks to my beloved wife, Maria de Jesus, and my lovely and extraordinary daughters Ileri, Carolina, Belen, y Sofia. My special gratitude to my sister Laura Teresa for providing a sympathetic ear and advice. I must thank my father, who has been a constant source of inspiration. He provided me with values that rank high in morals and character. I give my deepest love to my mother, Teresa Lara.

Thanks to my brother-in-law Jaime Reyes, to the kids Jaimito, Erik and to my niece Yara. Thanks for your love, encouragement and support throughout my PhD study and life.

I'm grateful to Professor Richard Bibb and Dr Patrick Pradel from Loughborough University in the U.K, to Professor Alberto Rossa from the Universidad Panamericana and MSc. Alejandro Briseño from the Universidad de Guadalajara, in Mexico.

Thanks to my fellow graduate students at the Institute of Design, Robotics and Optimization (iDRO) at the University of Leeds and the friends I made during this journey. A special thanks to Michelle Byrne for her support.

Finally, thanks the Mexican, National Council for Science and Technology (CONACYT) for facilitating my scholarship.

Abstract

Product development is a business process organisations use to introduce technological advancements to their products and services. Within this process, engineering design decisions define the architecture of the designed product, which, in turn, governs the structure of the product development process that ensures the quality of the delivered product. Integrating product development and engineering design processes results in a product development system characterised by networks of activities related to each other by feedback loops. Design iteration and rework are two kinds of feedback loop in product development systems. Design iteration is a form of positive feedback loop that contributes to the quality of the designed product. Rework, on the other hand, is a form of negative feedback loop that increases project duration and cost. Understanding feedback loops in product development is fundamental for effective design management and so the delivery of products on time and within budget.

This research contributes a conceptual framework for (a) understanding and (b) simulating the impact of feedback loops in engineering design. The framework enables the integration of two kinds of a simulation model. The first is a discrete event model that reflects the product architecture and its influence on the product development process structure, including potential rework feedback loops. The second is an agent-based model that reflects the social facets of design activity and communication behaviours within design teams, including design iteration. In this way, two kinds of feedback loop are captured: rework in the discrete event model and design iteration in the agent-based model. An engineering design case study was used to validate the conceptual framework.

This thesis takes a socio-technical systems perspective on engineering design. The conceptual framework includes relevant characteristics for simulating feedback loops in engineering design. The product architecture, which identifies individual parts that need to be designed and infers the development process structure, can be used to derive a design process workflow and so a discrete event model. In parallel, social interactions (actions, states, and behaviours) between designers are used to inform an agent-based simulation model of the design activities for each of the parts in the product architecture. Finally, the framework defines interaction points between the product development process and design activities which inform interplays between the two kinds of simulation models.

Table of Contents

A Conceptual Framework for Simulating Feedback Loops in Engineering Design	i
Acknowledgements.....	iii
Abstract.....	iv
Table of Contents	v
List of Figures	viii
List of Tables	xi
Chapter 1: Introduction.....	1
1.1 Product development systems	1
1.2 Feedback loops in product design and development systems.	3
1.3 Process models of product development	5
1.4 Product architecture.	6
1.5 Simulation modelling	7
1.6 Problem definition.....	9
1.7 Objectives	10
1.8 Thesis outline	11
Chapter 2: Literature review.....	12
2.1 Product development systems	13
2.2 Product development processes	15
2.3 Engineering design process	17
2.4 Process models of product development	17
2.5 Design iteration	20
2.5.1 Design iteration approaches.....	20
2.5.2. Micro, meso and macro iterations	27
2.5.3 Mental iteration and iteration of the design task	27
2.5.4 Modelling approaches to design iteration.....	28
2.6. Rework	29
2.6.1 Causes of rework	29
2.6.2 Mitigation of rework	31
2.7. Feedback loops.....	34
2.7.1 Vicious circles in organisations	35
2.8. Simulation models of product development process.	36
2.8.1 Computational models in engineering design.....	36
2.8.2 Simulation of iterative process	37

2.8.3 Simulation of engineering design teamwork.....	37
2.8.4 Simulation of the rework.....	38
2.9 Multiparadigm simulation modelling	39
2.10 Conceptual models.....	40
2.11 Conclusions	46
Chapter 3: Research methodology	50
3.1 Action design research (ADR).....	51
3.1.1 The four stages of ADR.....	51
3.2. ADR in this research	53
3.2.1 ADR Stage: Problem formulation	54
3.2.1.1: Conceptualise the research opportunity.....	54
3.2.1.2 Formulate initial research questions.....	54
3.2.1.3 Cast the problem as an instance of a class problem.....	55
3.2.1 Identify contributing theoretical bases prior to technological advances.....	56
3.2.2 ADR Stage: Building intervention and evaluation.....	57
3.2.2.1 Discover the initial knowledge-creation target.....	57
3.2.2.2 Execute building intervention-evaluation (BIE) cycles.....	58
3.2.2.3 Asses the need for additional cycle repeats.....	59
3.2.3 ADR Stage reflection and learning	60
3.2.3.1 Reflect on the design and redesign during the project.....	60
3.2.4 ADR Stage: Formalisation of learning	60
3.2.4.1 Articulate outcomes as design principles.....	60
3.2.4.2. Sub-task: Generalisation of design principles.....	61
3.4.2.3 Sub-task: Formulation of design principles.....	64
3.3 Case study	65
Chapter 4: Conceptual framework development.....	72
4.1. Problem identification	73
4.1.1 What is the problem?	74
4.1.3 Whose problem are we addressing?.....	75
4.1.4 Other actors	76
4.2 System identification	76
4.3 System conceptualisation.....	78
4.4 Model formalisation	83

4.5 Logical model	90
4.6 Summary.....	93
5.1. Implementation.....	110
5.1.2 Identification of MS methods	113
5.1.4 Relationship definition.	115
5.2.1 Discrete events modelling	123
5.2.2 Agent-based.....	125
5.2.3 Hybrid Simulation, Discrete events+Agent-based	130
5.3 Validation	133
5.3.1 Validation of the Discrete Events Model.....	133
5.3.2 Validation of the Agent Based Model	134
5.3.3 Validation of the hybrid simulation.....	136
Chapter 6: Conclusions	162
6.1 Research Contribution.....	163
Objective No 1.....	163
Objective No 2.....	165
Objective No.3.....	166
Objective No. 4,.....	167
Objective No.5,.....	168
6.2 Research Limitations.....	169
6.3 Future Work.	170
List of References	172

List of Figures

Figure 2.1.1.1 The NPD intra and extra-organisational context, adapted from de Weerd-Nederhof (2001).	14
Figure 2.2.1.1 Generic new product development process based on (Ulrich and Eppinger, 2012).	16
Figure 2.5.3.1 A cognitive activity model of conceptual design, from Jin and Chusilp (2006).	28
Figure 2.6.2.1 Summary main approaches on the mitigation of rework in the literature from table 2.6.2.1	34
Figure 3.2.1.1 Cross-gate's product development and architecture design and three in-stage iteration feedback loops.....	56
Figure 3.2.2.1 Abstraction level identification Djanatliev and German (2013)	59
Figure 4.1.1 Chapter layout	72
Figure 4.3.1.1 Ontology classes and instances identification.....	80
Figure 4.3.1.2 System conceptualisation using an ontological approach, based on Otte et al. (2019).....	82
Figure 4.4.1.1 Communication patterns between the actors in model narrative.	90
Figure 4.5.1.1 Conceptual framework for simulating feedback loops in engineering design.....	92
Figure.4.6.1.3.1 Scrum sprint ontology.	101
Figure 4.6.1.5.1 Sprint process narrative, based on Mvulane, (2020).....	105
Figure.4.6.1.5.3 Conceptual framework adaptation for the simulation of agile sprint processes.	107
Figure 5.1 Chapter layout.	109
Figure 5.1.1.1 UML activity diagram for the handlebar design process narrative.	112
Figure 5.1.2.1 Identification of abstraction levels, simulation approaches and methods adapted from Djanatliev and German (2013).	114
Figure 5.2.1.1 Discrete events simulation model, implemented in Anylogic 8	125
Figure 5.2.2.2 UML State chart diagrams for the agent-based conceptual model.....	127
Table 5.2.2.1 Initial parameters for agent-based simulation.....	128
Figure 5.2.2.3 Agent-based simulation diagram implemented in Anylogic 8	129

Figure 5.3.1.1 Time plots showing results of the discrete events simulations.	134
Figure 5.3.2.1 Time plots showing the results for agent-based experiments.	135
Figure 5.3.3.1 a,1 average time per part designer	138
Figure 5.3.3.2 a,2 average number of events per run.	138
Figure 5.3.3.3 a,3 average time per part designer	139
Figure 5.3.3.4 a,4 average number of events per run.	139
Figure 5.3.3.5 a,5 average time per part designer	140
Figure 5.3.3.6 a,6 average number of events per run.	140
Figure 3.5.5.8 b,2 average number of events per run.	141
Figure 3.5.5.7 b,1 average time per part designer	141
Figure 5.3.3.9 b,3 average time per part designer	142
Figure 5.3.3.10 b,4 average number of events per run.	142
Figure 5.3.3.11 b,5 average time per part designer	143
Figure 5.3.3.12 b,6 average number of events per run.	143
Figure 5.3.3.13 c,1 average time per part designer	144
Figure 5.3.3.14 c,2 average number of events per run.	144
Figure 5.3.3.15 c,3 average time per part designer	145
Figure 5.3.3.16 c,4 average number of events per run.	145
Figure 5.3.3.17 c,5 average time per part designer	146
Figure 5.3.3.18 c6 average number of events per run.	146
Figure 5.5.1 Simulation model diagram made in Anylogic 8.	153
Figure 5.5.2 Average time to complete a sprint (Scenario 1)	154
Figure 5.5.4 Average time to complete a sprint (Scenario 3)	155
Figure 5.5.3 Average time to complete a sprint (Scenario 2)	155
Figure 5.5.5 Average of accepted and not accepted designs (Scenario 1).	156
Figure 5.5.6 Average of accepted and not accepted designs (Scenario 2).	157
Figure 5.5.7 Average of accepted and not accepted designs (Scenario 3).	157
Figure 5.5.8 Average design jobs vs Sprints per run (Scenario 1)	158
Figure 5.5.9 Average Design jobs vs Sprints per run (Scenario 2)	159
Figure 5.5.10 Average Design jobs vs Sprints per run (Scenario 3)	160
Figure.4.6.1.3.1 Scrum sprint ontology.	
Figure 4.6.1.5.1 Sprint process narrative, based on Mvulane, (2020).	105

Figure 5.5.1 Simulation model diagram made in Anylogic 8.....	153
Figure 5.5.2 Average time to complete a sprint (Scenario 1).....	154
Figure 5.5.3 Average time to complete a sprint (Scenario 2).....	155
Figure 5.5.4 Average time to complete a sprint (Scenario 3).....	155
Figure 5.5.5 Average of accepted and not accepted designs (Scenario 1).....	156
Figure 5.5.6 Average of accepted and not accepted designs (Scenario 2).....	157
Figure 5.5.7 Average of accepted and not accepted designs (Scenario 3).....	157
Figure 5.5.8 Average design jobs vs Sprints per run (Scenario 1).....	158
Figure 5.5.9 Average design jobs vs Sprints per run (Scenario2).....	159
Figure 5.5.10 Average Design jobs vs Sprints per run (Scenario 3)...	160

List of Tables

Table 2.5.1.1 Design Iteration perspectives in literature based on Wynn and Eckert, 2017. Figures at the bottom of the table represent the percentage of studies that addressed each theme.	22
Table 2.5.1.2 Quantity of publications in different perspectives between the years of 1962 to 2020.	26
Table 2.6.2.1 Relevant frameworks developed for the mitigation of rework based on Dullen et.al, (2019).	32
Table 2.11.1 Hybrid traditional and agile project management approaches from Reiff and Schlegel (2022).....	43
Table 3.1.1.1 Tasks in each stage, adapted from Cronholm and Göbel (2022).....	53
Table 3.2.4.2.2 Types of value assessment relationships from Mykoniatis and Angelopoulou (2020).....	64
Table 4.4.1.1 Model narrative.	89
Table 4.6.1 Summary of the Scrum sprint events, roles and artefacts based on Mvulane, 2020).....	98
Table 4.6.1.4.1 Traditional product development process vs scrum sprint method.	102
Table 4.6.1.5.1 Variables identification.....	104
Table 5.1.4.1 Categories and types of relationships, based on Mykoniatis and Angelopoulou (2020).....	116
Table 5.2.1.1 Initial parameters for the discrete events model.....	124
Table 5.2.2.1 Initial parameters for agent-based simulation.....	128
Table 5.2.3.1 Summary of the interaction points identified.....	132
Table 5.3.3.1 Validation experiment parameters.	137
Table 5.5.1 Summary of actors, relationships, interactions, and states.....	150
Table 5.5.2 Simulation initial parameters. Description of blocks form Anylogic 8.....	151
Table 5.5.3 Experiments design specification.	154

Chapter 1: Introduction

Product development processes codify the ways in which manufacturing organisations deliver new products to markets in response to customer demands for requisite quality and strategic priorities to release products as quickly and cost-effectively as possible. These processes involve a series of stage gates where decisions to proceed or not are made. Such decisions are important in managing product development processes because they ultimately drive performance against time and cost indicators (Tapia et al., 2021). On the other hand, engineering design processes lie between the stage gates and, through the creativity and capability of engineering design teams, significantly impact the quality of products delivered to customers. A key challenge for design managers lies in balancing the dynamic nature of product development processes resulting from positive and negative feedback loops within them.

This research explored the potential value of advanced computer simulation in understanding the impact of two types of feedback loop (design iteration and rework) on the performance of product development systems. In the long term, this research has the potential to inform a new generation of engineering management solutions.

1.1 Product development systems

Pessôa and Trabasso (2017) assert that product development systems are organisational systems integrated by individuals and teams. They use resources and technologies to perform activities and processes to transform inputs from market opportunities and customer requirements into outputs in the form of technical descriptions or material objects. In the same context de Weerd-Nederhof (1997) explain that product development systems are characterised by intra and extra-organisational contexts. The extra-organisational context includes entities influencing the system without affecting its functions (e, g., other organisations, like competitors, supply chain partners, or government entities such as regulatory authorities). On the other hand, the intra-organizational context is integrated with other business functions (such as marketing, sales, procurement, engineering, and manufacturing) influencing the system's performance through its interactions.

Product development systems articulate a market opportunity and execute a product development process (Pessôa and Trabasso, 2017). When receiving inputs from its

environment, a product development system generates outputs from technological advancements, creative ideas, and products or process designs (de Weerd-Nederhof, 1997). The execution of the product development process, including activities to produce, sell, and deliver the developed product, is often planned sequentially to form the workflow (Montagna and Cantamessa, 2017), where engineering design teams carry out the development phases sequentially, separated by stage-gates reviews (Shepherd and Ahmed, 2000). These stage gates include predefined checkpoints (gates) containing deliverables for each functional area, concluding the process when all the required information to support production, sales, and delivery of the developed product has been created and communicated (Ulrich and Eppinger, 2012).

Although the product development process is a chronological succession of tasks and activities, in real life, the process entails a series of planned and unplanned iterations that cause interruptions due to critical design issues and unplanned or discrepant communications. From this perspective, the product development system can be seen as a social network where engineering designers and stakeholders interact to find a design solution (Montagna and Cantamessa, 2017). Uncertainty is a relevant characteristic manifested in the lack of consistent information and multiple conflicting interpretations (Pessôa and Trabasso, 2017).

Engineering design processes reside at the core of new product development systems, materialising opportunities identified by marketing and translating user needs, requirements, constraints, and specifications into a technically feasible and usable solutions. They are technical processes through which innovations are developed and embedded in products providing the structures for stages of product development (Tapia et al., 2021). Engineering design processes are systematic and intelligent processes where designers generate, evaluate, and specify concepts for devices, systems, or processes (Dym et al., 2005) for the whole product and its components. Engineering design processes are directed by the decisions made by individuals in design teams (Wallace and Ahmed, 2003) where effective management of communications, negotiation, and coordination mechanisms used by actors influence the outcome and progress of the design work (Hoegl and Weinkauff, 2005; Maier, A.M. et al., 2007). Engineering design activities of acquisition and provision of information are widely recognised. King (1994) asserts that, in engineering design, most design activity consists of creating, transferring, or disseminating information. For example, in one study engineering designers spend 24% of their working time in activities related to information behaviours (Marsh, 1997). Meho and Tibbo (2003) identified four stages of information seeking for

engineering designers: (1) searching for information and identifying relevant sources, (2) accessing and acquiring information from those information sources, (3) processing and analysing the obtained information, and (4) finalising the search process.

This research used all four of these information-seeking stages to reflect team-based process activities: Requesting, Answering, Receiving, and Evaluating information. These process activities maps de Meho and Tibbo (2003) stages, the searching for information and identifying relevant sources, maps with requesting information; Accessing and acquiring corresponds to answering and receiving information. Processing and analysing information, maps with the evaluation of information stage.

1.2 Feedback loops in product design and development systems.

Feedback loops are an essential concept in social and organisational theory that enhance understanding of relationships between a system's past and current states (Tsoukas and e Cunha, 2017). The literature related to feedback loop uses the term to describe activities aimed to reduce a gap between a perceived and future state of a system. A different trend in the literature suggests that in feedback loops, the output of a process influences its input directly or indirectly at some point in time. Feedback loops are associated with the evolution of a system over time when they are related to control and stabilisation and with the improvement or with the decline of a process or behaviour when they are associated with virtuous and vicious circles (Masuch, 1985).

Edgeman et al. (2020) assert that within organisations, patterns or cycles of behaviour identified as enterprise routines or habits that produce predictable poor to negative results might be regarded as a vicious circle. A vicious circle is a self-propagating complex chain of events with failures or negative consequences at one stage that generate increasingly serious failures or negative effects at each subsequent stage. On the other hand, behaviour patterns producing positive results can be regarded as a virtuous cycle, a self-propagating complex chain of events with positive consequences at one stage generating positive outcomes at each subsequent stage (Edgeman et al., 2020).

During the development of the design activity, new information and constraints emerge, and design requirements change. These changes lead designers to revisit and re-evaluate previous design decisions (Wynn and Eckert, 2017) and adopt new ones, generating new activities and information feedback cycles. These iterative cycles are positive feedback loops that contribute to the quality of design within the

design stage, or negative feedback loops when the new information requires modifications on previous activities considered already finished from an earlier phase.

Design iteration is a critical feature in the design process that enables the progressive generation of knowledge, concurrency, and integration of necessary changes (Wynn, Eckert 2017). With the systematic exploration and understanding of the design problem's complexity (Le, H.N. et al., 2010), iterations are natural means that help designers and engineers better understand the design problem and solutions (Eckert et al., 2014). It contributes to the incremental completion of the tasks with different information levels and improves quality, but eventually also adds time to design activity and, hence, the development process. The term iteration can refer to both broader loops, such as the successive model releases of a product, or narrower loops, e.g., when mathematical techniques are used for engineering optimisation processes (Safoutin and Smith, 1996).

In the literature, design iteration is defined as the repetition of an activity to generate meaningful information to represent and refine a design solution towards a desired final state. Designers adopt behavioural iteration by dividing a problem into pieces and performing similar patterns of design activity on each part (Costa and Sobek, 2003). In contrast, Jin and Chusilp (2006) noted that the iteration of the design task is the repetition of a task from a design team due to new information or because of previous iterations and a mental iteration is characterised by the cognitive activities of designers when performing the design tasks.

Engineering designers recognise iteration as an investigative tool that enables advanced knowledge and a better understanding of the design problem and solution (Dorst and Cross, 2001). They are essential learning cycles that allow continuous knowledge gain, mitigating uncertainty and ambiguity (Meboldt et al., 2013). On the other hand, from the managerial perspective, design iteration needs to be fitted into different project planning strategies, such as parallel, sequential, or independent task cycles, to develop planning models that incorporate the feature in the overall development process. From the management point of view, iterations are expensive exceptions, costly, and time-consuming. For instance, Meboldt et al. (2013) asserts that in-stage iterations are iteration within a stage and is expected that there is no impact previous stages. While cross-gate iterations are iteration affecting previous stage decisions, influencing the project cost, time, and quality.

Rework is considered unnecessary work and delays caused by redoing a process or activity not adequately implemented the first time (Love, P.E., 2002). Rework

iteration that occurs with the repetition of the activity at the same level of abstraction in the same object is generally used to correct an error (Costa and Sobek, 2003). Furthermore, the analysis of the literature suggests that rework is a result of the information dynamics of the new product development process, caused by the inadequacy of the information due to changes in requirements, poor decisions, defective outputs, or changes in implementation that alters work previously done (Smith, R., P. and Eppinger, 1997a). While design iteration is in-stage, decisions do not affect other stages (Meboldt et al., 2013). Rework is a cross-gate iteration where decisions affect decisions made in previous stages affecting project time and cost.

1.3 Process models of product development

Most of the research literature on engineering design has established that adequate management of the design and development processes at individual, team, or organisational levels is a key feature for developing acceptable designs and reducing the problems related to their development (Wynn et al., 2019).

The organisational activities in engineering design and development processes related to individual activities and their context (micro-level), including those associated with the flow of tasks and design progression (meso-level), and those related to project/programme and contextual considerations (at the macro-level), require significant coordination to manage the complex dependency structures of the product development processes (Wynn et al., 2019). To tackle this rising complexity, a better understanding of the design and development processes, including tools, and methods to support design management, is required. Modelling and simulation allow the analysis and prediction of systems and processes behaviours that are too costly, dangerous, or time-consuming to understand through actual conditions (Eckert et al., 2019).

A model is a conceptual representation of an object, process, or system that can be verified, analysed, and manipulated for a particular purpose. They do not exist in the real world, and their construction provides understanding, control, and learning about the system they represent (Smith, R., P. and Morrow, 1999).

Early process models of product development are mainly derived from traditional project management methods. By listing the activities to perform and identifying dependencies, planners identified the critical paths and duration and then explored opportunities for duration time improvement (Browning et al., 2006). Such model approaches were, for instance, the program evaluation review technique (PERT) and critical path method (CPM), where the development process was viewed as a

network of discrete activities with stochastic duration and sequential relationships (Pritsker, 1966; Ritchie, 1972). On a different note, from the complex system perspective, product development is viewed as a system where the combined attributes and interactions of people, products, and processes generate non-linear behaviours (Chiva-Gomez, 2004; McCarthy et al., 2006).

Process models are means for understanding and interacting with products and processes (Eckert and Stacey, 2010). They enable visibility and transparency to the workforce, providing, in some cases, process-related best practices and a baseline for process management. Process models of product development allow change analysis and support the understanding of the modelled systems (Browning et al., 2006). They provide insights at different levels depending on the application and interpretation, emphasising other process elements. Process models can offer different terminology and visual representations (Wynn and Clarkson, 2017) of the complex interrelations of the complex product development processes.

Process models of product development can be identified into two categories: descriptive models, which are those models used to record what happened in a process; and prescriptive models, which are those that direct what should be done in a particular type of product or process (Browning et al., 2006). In the literature, there are also classifications for stage-based models, activity-based models, and design processes, the latest identified as problem-oriented and solution-oriented models (Wynn and Clarkson, 2005).

1.4 Product architecture.

The complexity of new products arises due to the uniqueness and complexity of the components and subsystem interactions. New products are often developed by autonomous design teams distributed within multiple firms and manufactured by complex supply chains.

Product architecture is used to assign a product's functional elements to physical building blocks to determine what they do and their interfaces (Ulrich and Eppinger, 2012). Product architecture decisions allow detailed design and testing to be assigned to teams, individuals, and suppliers, allowing the development of different product portions to be conducted simultaneously. The fundamental decisions made when the system architecture is defined (Jankovic and Eckert, 2016) determine the success of a new product in the market. At this stage, the completed design requirements definition and functional and physical configurations are established to define the product development tasks.

The mapping of the functional design elements to physical parts makes possible the definition of the interfaces among the interacting components (Ulrich, 1995; Sharman and Yassine, 2004), providing a description of system boundaries and the selection of fundamental solution principles for the overall design.

Product development processes start with articulating a market opportunity and are integrated into six stages: planning, concept development, system-level design, detailed design, testing and refinement, and product ramp-up (Ulrich and Eppinger, 2012). There is an evaluation gate between each stage where the deliverables must be passed to proceed to the next (Tapia et al., 2021). The actions and activities within these processes are, in most cases, intellectual and organisational rather than physical and include developing information and formulating specifications, concepts, and design details. The product development process concludes when all the required information to support production and sales has been created and communicated (Ulrich and Eppinger, 2012).

The product architecture begins to emerge during the conceptual design stage (Ulrich and Eppinger, 2012). The design process translates user requirements into a structural description of the arrangement and systems of the components in the product, incorporating the understanding and sometimes explicit description of the functions and behaviours the product will carry out (Jankovic and Eckert, 2016). It is important because it impacts the structure of the product development process and so the design of each design unit needed to develop the resulting design.

1.5 Simulation modelling

Simulation modelling is used in this research process because it enables the evaluation of products and systems before they exist, supporting decision-making (Yin, C. and McKay, 2018). Simulation models aim to explain correlations between process variables measured at one point in time but also include explicit representations of processes that work in the social world. Simulation modelling usually starts with a problem that needs to be solved or the need for a better understanding of a situation, often addressing issues that frequently present a need for more knowledge about real-world systems, their behaviour, or their response to a particular intervention (Nikolic and Lukszo, 2013). Simulations also serve as a laboratory for experiments, less complex, costly, or dangerous to perform than actual life experiments (Nikolic and Ghorbani, 2011). The ultimate objective of a modelling exercise is to gain insights into the system, but not necessarily to produce numbers. Simulation models serve as a tool to improve understanding of the dynamics of the

whole system or subsystems by exploring possible states or finding states to be approached or avoided through what-if scenarios (Nikolic and Lukszo, 2013).

Simulation model construction involves carefully established steps to produce a reliable representation of the real world. Those steps may vary depending on the modeller's background, simulation paradigm, or methodological approach to the problem. Simulation modelling designs must reflect real-world systems' outcomes, and simulation techniques develop experiments that allow the understanding of the system's performance under different operating conditions to evaluate management strategies or decision-making processes (Nikolic and Lukszo, 2013).

Borshchev and Filippov, (2004) assets that simulation model construction distinguishes between analytical or static models and dynamic or simulation models. The analytical model's outcome depends on the input, while dynamic modelling uses a set of rules that define how the modelled system will change in the future (Borshchev and Filippov, 2004). There are three standard simulation approaches to representing the business process, Borshchev, (2013) Points that, system dynamics, use a high abstraction level and thinks in terms of stocks, flows and feedback loops. On the other hand, discrete events modelling is a process-oriented approach where the system is represented as a sequence of operations performed over activities, In the agent-based modelling, individual objects interact with other individual objects and with the environment (Borshchev, 2013).

The considerations for the simulation of feedback loops in engineering design processes include: (1) The design structures of the system. Identifying the design product architecture (Maier, J.F. et al., 2014) and decomposing the system into its components, distinguishing interfaces, and interactions. (2) The characteristics of social interactions of product development teams (Montagna and Cantamessa, 2017) are influenced by information (Robinson, 2010) and communication patterns during the design process. (3) The interrelationships between product development and engineering design processes, where the development process is depicted as a series of sequential and discrete events driven by a chronological progression of tasks with a series of decision gates (Artmann, 2009), and the engineering design process frameworks for the stages including the development of designs for the whole product and its parts (Tapia et al., 2021). [4] Product development and engineering design interactions determine the system's ability to process design requests. The design process governs the time to complete a design task, and product development influences the ability to complete a product design.

1.6 Problem definition.

Pessoa and Trabasso, (2017) argue that a product development system is a complex socio-technical system integrated into a multidimensional network of interconnected processes where feedback loops traverse through its hierarchical levels. The product development system is integrated with three relevant system elements: people, process, and product.

When the product development system receives inputs, information, or goods through its boundaries, it processes those inputs into outputs. It delivers them back to the environment in the form of products or services (Pessoa and Trabasso, 2017). Ulrich and Eppinger (2012) The product development processes performed within the product development system are composed of intellectual and organisational activities, developing information, formulating specifications, concepts, and design details. These processes conclude when all the required information to support the production and sale of the developed product has been created and communicated (Ulrich and Eppinger, 2012).

Feedback loops in product development systems, which arise from highly interconnected processes (Pessoa and Trabasso, 2017), result from engineering design information processes and coordination and collaboration activities among designers and design teams (Wynn and Maier, 2022). Feedback loops are small or large recursive cycles that characterise relationships and iterations (Kline and Rosenberg, 1986) between product development processes and participants, caused by new information, emerging constraints, and changes in design requirements during the design activity. These changes lead designers to revisit and re-evaluate previous design decisions (Wynn and Eckert, 2017) and adopt new ones, generating new activities and information feedback cycles. These iterative cycles are positive feedback loops (virtuous circles) when information is adequate, correct, and delivered on time, contributing to design quality within the design stage. On the contrary, negative feedback loops (vicious circles) result from new information or result from inadequate, inaccurate, or miss-coordinated information, which makes modifications necessary to previous activities that have already been completed in an earlier phase.

The study of new product development systems and processes as complex socio-technical systems uses simulation tools to explore the possible outcomes of interactions between the system elements (Perišić et al., 2016). These analyses include technical aspects such as technology, infrastructure, processes, and social features (Clegg et al., 2017). However, the approaches of simulating product

development processes and design teams often prioritise certain parts of the system. For instance, simulation models that estimate technical performance tend to ignore social processes, while models exploring social aspects may omit tasks, resources, or project details. Ideally, a model that enables the study and measurement of both intangible individual and team aspects, as well as tangible aspects, could provide a more comprehensive view of the system's performance (Škec et al., 2017).

A realistic representation of the product development process must reflect the organisational and process characteristics that mirror the product architecture, which ultimately influences the process and organisational structure of the development activity. A simulation model trying to create realistic simulations with a comprehensive view of the engineering design process must consider the technical aspects of the product development, such as the design structure of the product, the linear logic of the product development process, along with the social aspects of design teams' communication patterns and feedback loops. This research focuses on identifying key elements that enable the construction of a simulation framework to capture the engineering design processes, technical processes, and social aspects.

Thus, the research problem identified is the need for a comprehensive simulation framework for feedback loops in engineering design that considers both technical and social aspects of the product development process. The overarching goal of the research is to identify critical elements that enable the construction of such a framework, allowing for more realistic simulations and a better understanding of the system's performance.

1.7 Objectives

This research develops a conceptual framework for simulating feedback loops in engineering design that takes into account both technical and social aspects of the product development process. It identifies key elements that enable the construction of a simulation framework that captures the engineering design processes, technical processes, and social aspects of the product development process. In so doing, it allows a realistic representation of the product development process, reflecting the organisational and process characteristics that mirror the product architecture, which ultimately influences the process and organisational structure of the development activity.

To identify the key characteristics of product development systems that impact system performance the following objectives were pursued.

- 1) To identify key technical and social aspects of product development processes for the simulation of feedback loops in engineering design.
- 2) To identify critical characteristics of feedback loops that influence the performance of product development processes.
- 3) To design and develop a conceptual framework for simulating feedback loops in engineering design that incorporates the identified critical elements for the implementation of more realistic simulations of product development processes.
- 4) To implement an engineering design process case study for use in validating the framework.
- 5) To consider how such simulation models might be used to inform the management of product development systems.

1.8 Thesis outline

This thesis introduces a conceptual framework, which is then incorporated into a simulation model which combines two different simulation approaches to simulate feedback loops (rework and iteration) in engineering design.

Chapter 2 reports the literature review, which explored and assessed the literature around the product development systems, including approaches to process models of product development, product development systems as complex systems, feedback loops, iteration and rework, and computational models in engineering design. Chapter 3 introduces the overall research process used in developing the conceptual framework using Sein et al.'s (2011) Action Research Method (ADR), summarising the method's main stages and task principles, and detailing how the Action Design Research method was used to support the development of the conceptual framework for the simulation of feedback loops in engineering design. This chapter also includes details of the bicycle design case study in Section 3.3. Chapter 4 describes the conceptual framework for simulating feedback loops in engineering design, integrated with four stages: (1) The problem formulation stage, establishing the problem to be addressed. (2) The system identification stage, which consists of the inventory of the system. (3) The system conceptualisation stage, where the identified concepts are formalised. (4) The model formalisation stage, where the conceptual framework is deployed as a simulation model. Chapter 5 includes the simulation process implementation to evaluate the conceptual framework, showing the simulation models' results separately and the hybrid simulation results. Finally, Chapter 6 provides the conclusions, summarizes the knowledge contribution, and makes recommendations for future work.

Chapter 2: Literature review

The overall purpose for research in new product development is to support business organisations by improving the understanding of their processes. Theories, methods, methodologies, strategies, and models are developed, discussed, and published year by year in order to do so, in specialized journals including *Management Science*, *Research in Engineering Design*, *Design Studies*, *Journal of Mechanical Design*, and *Engineering Management*. Supporting this research dialogue, there are regular publications where academics and practitioners decompose the new product development process into its different elements and stages to understand and improve systems, process and interactions, and its impact in whole organisations, outcomes, and performance indicators.

The initial purpose of the literature review in this thesis was the identification of the relevant characteristics of design iteration in the context of product development processes. However, during the literature review, it became evident that design iteration is a result of the information interactions between design engineers and stakeholders, while searching for a design solution (Montagna and Cantamessa, 2017) at different hierarchical levels within a product development system (Pessôa and Trabasso, 2017). Furthermore, design iterations are influenced by the organisational and social characteristics of designers or design teams performing engineering design activities within the development processes. Consequently, the focus of this literature review chapter in this thesis is to examine the relevant characteristics of positive and negative design iterations in the context of product development systems. In turn, this enables the construction of a conceptual framework for simulating of feedback loops in engineering design. To do so, the following objectives were pursued in this chapter.

- 1) To identify key technical and social aspects of the product development process required for simulations of feedback loops in engineering design.

The literature review chapter begins by analysing the literature on product development systems, followed by the literature related to product development processes, including literature relative to models of product development and engineering design processes. It dedicates an extensive analysis to design iteration, and also revises the literature related to rework, causes, and mitigation, and includes literature related to feedback loops and vicious circles in organisations. The second section of the literature review is focused on simulation models of product development. It includes a review of the literature related to computational models of engineering design, the simulation of iterative processes, and the simulation of

engineering teams. Finally, the section reviews aspects of the simulation of rework cycles, modelling simulation, and conceptual models.

Section 2.1 addresses the new product development process system, followed by Section 2.2 which reviews the new product development processes. Section 2.3 discusses the process models of product development, while Section 2.4 looks at the research literature related to engineering design. Section 2.5 reviews the literature pertaining onto design iteration. Section 2.6 reviews the literature about rework and Section 2.7 reviews the literature pertinent to feedback loops. Section 2.8 presents aspects of simulation models of product development, before Section 2.9 reviews multiparadigm simulation modelling, and Section 2.10 finishes by reviewing conceptual models.

2.1 Product development systems

The product development system, from the organisational and management perspective, is considered a system of individuals and resources that: (a) use technologies, (b) perform activities and processes; (c) transform inputs in the form of perceived market opportunities into outputs in the form of products or services, to be delivered to its environment; and (d) when those outputs are considered useful by the environment, generate the revenue that enables the organisation to fulfil its goals.

De Weerd-Netherhof (1997) asserts that ,the new product development system receives inputs from its organisational context, produces a set of preliminary outputs in the form of technological advancements, creative ideas, product or process or designs, as well outputs in the form of technical, managerial or commercial decisions that are delivered into the system which are then transformed into measures to support the processes, technological advances, and the new outputs to the extra-organisational context .The product development extra-organisational context includes all the entities influencing the system without affecting its functions (e, g., other organisations, supply chain partners or government entities such as regulatory authorities). Finally, the intra-organizational context integrates other business functions (like marketing, sales, procurement, engineering, and manufacturing) that influence the system's performance (de Weerd-Nederhof, 2001), (figure 2.1.1.1) .

The product development systems articulate a market opportunity and execute a product development processes (Pessôa and Trabasso, 2017). The execution of the product development process includes the activities of production, sales, and delivery of the developed product.

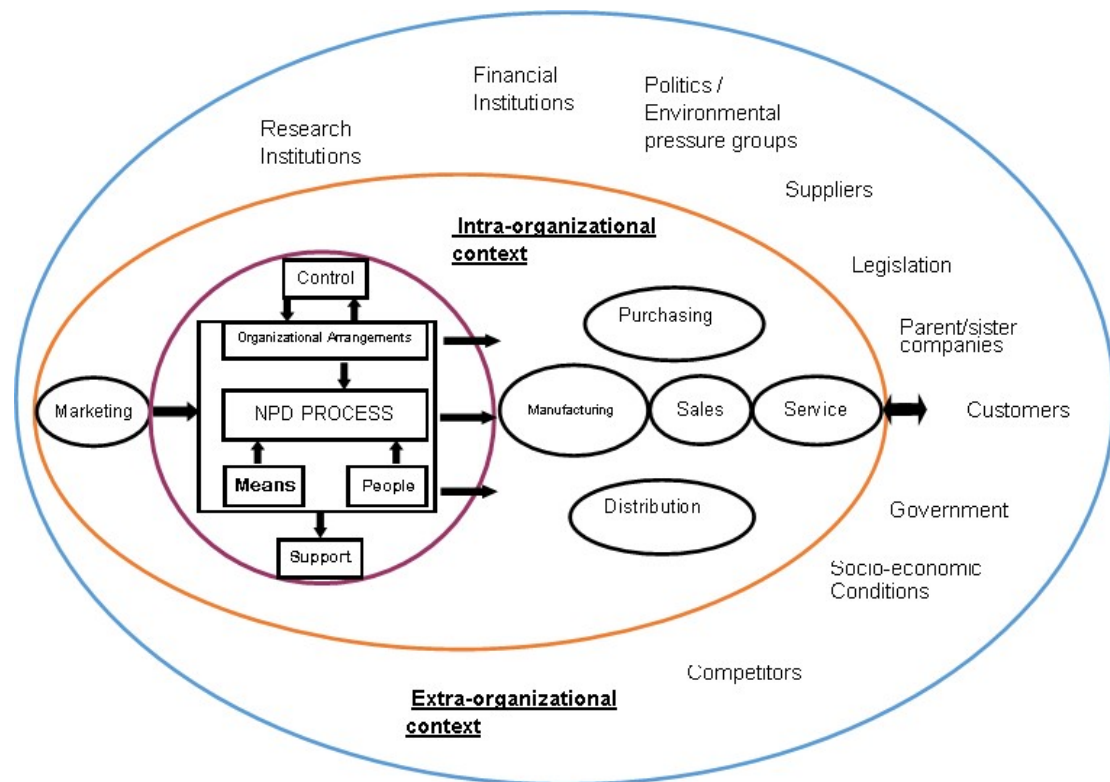


Figure 2.1.1.1 The NPD intra and extra-organisational context, adapted from de Weerd-Nederhof (2001).

For Rycroft and Kash, (1999) product development is a large, multidisciplinary, distributed, and networked system that cannot be embraced by a single group or organization. They assert that product development is an amalgam of the product, people, processes, and their interdependencies in each domain. Complex product development systems are multidimensional by nature, comprising product architecture, communication patterns (Yassine, 2018; Yassine, 2019), iterations and rework as key features that interact within the three domains of task quality, project schedule, and design teams. The modern product development processes require simultaneous and multiple group collaborations, producing and exchanging knowledge and information from different perspectives that overlap and interoperate simultaneously during the process to find an effective solution (Szejka et al., 2017).

Szejka (2017) analysed product development processes, suggesting the use of multi-perspectives, wherein a domain perspective, different specialists produce and share information to design and manufacture a product. Szejka et al. (2017) asserts too that the product development perspective includes three main phases: predevelopment, development, and post-development, and the subdivisions of the phases depend on the enterprise process development method or product

characteristics. Each phase has its proper constraints and specific information that impacts future or previous phases. In a phased process, the outcomes of a given phase serve as the inputs for the next phase. Therefore, any impacts that arise in a future phase are a direct result of the outcomes of the previous phase. On the other hand, if there is a change in a current phase, it may also have an impact on a previous phase that needs to be evaluated to assess any potential impacts (Szejka et al., 2017).

2.2 Product development processes

At the core of the new products' organizational context, the product development processes are strategies companies use as a competitive advantage (Krishnan, V. and Ulrich, 2001), delivering new products in response to market demands and strategic priorities. The product development process starts articulating a market opportunity and initialises the product development processes (Ulrich and Eppinger, 2012) The generic product development process is depicted as a sequential approach driven for the chronological progression of the development tasks, suggesting a series of stage-gates, where decisions to proceed or not drive the projects (Artmann, 2009)

The process can be divided into a set of stages with predefined checkpoints (gates) that contain deliverables for each functional area that must be approved to proceed to the next (see figure 2.2.1.1)The traditional product development process model prioritizes quality and key performance indicators like costs and time to market and it is a highly cross-functional process (Artmann, 2009). The product development process is integrated into six stages: (1) planning, (2) concept development, (3) system-level design, (4) detailed design, (5) testing and refinement, and (6) product ramp-up.

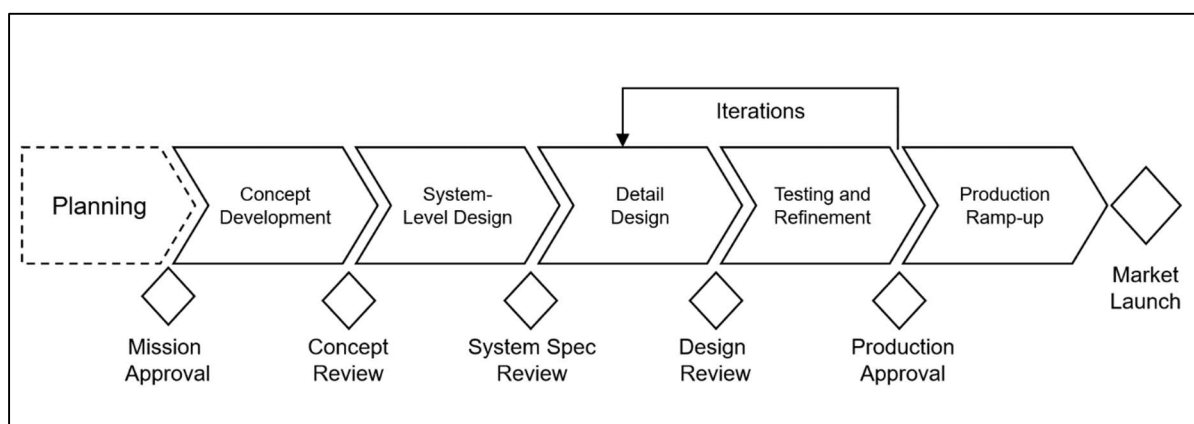


Figure 2.2.1.1 Generic new product development process based on (Ulrich and Eppinger, 2012).

Artmann (2009) explain that the planning stage entails the investigation of the potential market for a product, the exploration of possible architectures, manufacturing methods, and financial studies. After the formal approval of the project. During the concept development stage, alternative product concepts are generated and evaluated, and different system architectures are considered and defined for the overall system; in this stage the required tasks, budget, and constraints are explored and refined (Ulrich and Eppinger, 2012). The system-level design phase includes the system architecture's definition, which decomposes the product into subsystems and components to assign the necessary teams to develop each part.

Artman (2010) also asserts that the detail design stage is a highly parallel process where the development teams work simultaneously but separately. In this phase, the complete specification of the product, including geometry, materials, and tolerances of each part, is made. The testing and refinement stage includes the production of prototypes to determine if the product will work as designed. Finally, in the production ramp-up, the product reaches manufacturability, and after cycles of building, testing, and refinement iterations, the product arrives to the ramp-up stage to finally be launched to the market (Artmann, 2009). The production is manufactured using the intended production system, including training activities and refining production processes.

The product development process concludes when all the required information to support production and sales has been created and communicated (Ulrich and Eppinger, 2012).

On a different note, planning in product development is interpreted as a sequence of activities and workflows, that use a more or less sophisticated modelling approach focusing on the balance of the trade-offs between resource allocation and duration/cost of the process (Montagna and Cantamessa, 2017), neglecting the organizational issues that affect the process behaviour.

The social network with actors interacting directly to find a design solution is influenced by the social, behavioural and communication patterns of the participants (Montagna and Cantamessa, 2017), which are affected themselves by cultural, expertise and experience differences (Bucciarelli, 1988). Consequently, the outcome and progress of the design work depends on the effective management of communication and coordination mechanisms used by the actors, so product

development can be seen as social system set in a technical base (Whitworth, 2011).

2.3 Engineering design process

The engineering design processes provide the frameworks for the stages of product development projects that include developing designs for the whole product and its parts, include the technical processes through which innovations are developed and embedded in products. The performance of those process is governed by the creativity and capability of the engineering design teams that prioritise technical quality to fulfil all design requirements, but also carry out activities in the context of tight deadlines. (Tapia et al. 2021).

Through a systematic and intelligent process, engineering designers generate, evaluate, and specify concepts for devices, systems, or processes, in form and function to achieve clients' objectives' or users' needs while satisfying a specified set of constraints (Dym et al., 2005). Accordingly to Maier, A.M. and Störrle (2011) the main characteristics of the engineering design process are its complexity, its iterative nature, and the ill-defined of its problem formulation.

2.4 Process models of product development

Product development uses the process modelling perspective as means for capturing and describing, patterns and behaviours, to support problem-solving, decision-making, and as common platform for communication (Maier, A.M. and Störrle, 2011) Process models are useful means of understanding and interacting with both products and processes (Eckert and Stacey, 2010). Their construction helps the team focus and provides visibility and transparency to the workforce, indicates process-related best practices, and provides a baseline for process management, allowing change analysis and supporting the understanding of complex processes (Browning et al., 2006).

The most commonly used design process models in practice are based on flowchart diagramming, typically including activities connected by information flows and logic gates that determine the sequences in which tasks are attempted (Wynn and Clarkson, 2017). Another popular approach to this kind of modelling is the Business Process Modelling Notation (BPMN), and similarly Event-driven Process Chain (EPC), which offer the advantage of easy-to-interpret diagrams and robust software tools available for the creation of large-scale models (Wynn and Clarkson, 2021). Amigo et al, (2013) performed a systematic literature review with the aim to provide a

state-of-the-art picture about process modelling methods and propose a detailed classification based on their purposes. They found that there is not an agreement on the definition on the term modelling method, as the terms normally used are frameworks approach, techniques, languages, and views. They concluded that the purposes of models of product development are from high abstraction levels showing “flow of data information”, to less abstract levels that “define/show activities sequence (Amigo et al., 2013). Wynn and Clarkson,(2007) performed a literature review on the principal models in design and development, providing a taxonomy for procedural and analytical models of product development and design. They also identified categories for abstract models and models related to the management research operations. Their taxonomy also categorized the utilization of models at a micro-level, which are models that focus on individual process steps and their immediate contexts; meso-level models, which are focused on end to end flows of tasks as the design progressed; and macro-level models, that focus on project structures, or design processes in context.

Models provide insights at different levels depending on their application and the interpretation, and each emphasises different elements of the process, and offers different terminology and visual representations (Wynn and Clarkson, 2017). Browning et al., (2006) explains that process models are used to develop understanding, or for planning by determining what needs to be done and when; to prescribe a procedure to be followed, or to predict a possible a process behaviour. Descriptive process models attempt to capture tacit knowledge about how work should be, describe key features of the "as is" reality, and achieve their purpose when they provide a valid understanding of the target system (Browning et al., 2006). The prescriptive process models tell people what to do and how to do it, as a standard process or procedure accompanied by a mandate to follow it strictly (Browning et al., 2006). In prescriptive models, the relation between model and target is deontic, by defining what should be done, the model precedes its target (Eckert and Stacey, 2010).

Models are a central element of design methodologies, as they provide a consistent terminology and can be identified, as stages, (or design phases), activities, and strategies (Blessing, 1995). Gericke and Blessing (2011) identified, stages, design activities, and strategies, as follows. A stage is subdivision of a design process in relation to the state of the designed product, where every stage considers a time-consuming activity. A design activity is a finer-grained division than a stage, a subdivision of the design process associated with the individual problem-solving process. Strategy is defined as the sequence in which design stages and activities

are planned or executed. Those strategies provide ways to execute the design process, i.e., stepwise, cyclic, decomposing, iterative and abstracting /concretizing (Gericke and Blessing, 2011).

In a literature review, Wynn and Clarkson (2005) identified design stages or design activities, distinguishing between three dimensions for models of design processes. Firstly, classified as (1) stage-based models, (2) activity-based models, then followed by (3) design process models identified as problem-oriented, and solution-oriented. Wynn and Clarkson, (2005) also describe models with abstract approaches, including procedural approaches and analytical approaches.

On a different note, Chakrabarti and Blessing (2016b) provided an analysis of models' design in their "Review of Theories and Models of Design", categorising models of design into those initiated before this century and those initiated during this century. It is a chronological analysis of the design models in the engineering design research field, providing an analysis of design theories and how they differ from models of design. Chakrabarti and Blessing (2016a) reflect as well on how theories and models have become more widely known, and how their construction has become more rigorous, looking for validation using data and linked to practice.

Chakrabarti and Blessing (2016a) established that object models related to designing, such as scale models, CAD models, and sketches are *design models*. While models used to describe or prescribe how design and designing should be, and how those relate to practice or education may be called *models of design* (Chakrabarti and Blessing, 2016b). They stated as well, that a model could be a subset of a theory, where the theory provides a higher level understanding than a model does (Chakrabarti and Blessing, 2016b). Their anthology relates in most of the cases descriptive models, many of the ones initiated before this century are organized from the top-down perspective and around the product breakdown, in an inherited style from manufacturing and construction industries (Kruchten, 2002). While in some models initiated this century a top-down/bottom-up continuum across the development cycle (Kruchten, 2002) is becoming more evident.

The review of the literature suggests that since design processes consider different requirements and constraints within their context (Kohlberg et al. 2014), not all models can be relevant to every situation. The assumption of a logical and predictable order of activities in all design processes is wrong because there is not a sequence of operations which guarantee success, and therefore the adaptation of the process model according to the specifics of the problem is necessary (Gericke

and Blessing, 2011). The ability to manage this adaptation in one of most important skills of designers (Lawson, 2006).

2.5 Design iteration

Design iteration is recognized for having a ubiquitous character in the design process. By enabling the progressive generation of knowledge, concurrency, and integration of necessary changes (Wynn, Eckert 2017) with the systematic exploration and understanding of the design problem complexity (Le, H.N. et al., 2010), iteration contributes to the incremental completion of the tasks with different information levels and improving quality, but eventually adding time to design activity. It is recognized for engineering designers, as an investigative tool that enables advanced knowledge and a better understanding of the design problem and solution (Dorst and Cross, 2001). On the other hand, from the managerial perspective, design iteration needs to be fitted into different project planning strategies, such as parallel, sequential, or independent task cycles, to develop planning models that incorporate the feature in the overall development process.

An important trend in the engineering design research literature discusses design iteration, assessing theories, descriptions, or taxonomies. A number of researchers argue that design processes and large projects such as product development are iterative in nature and the importance of managing iteration has been well established (Wynn and Eckert, 2017).

2.5.1 Design iteration approaches

Design iteration has been analysed under several perspectives. In a recent literature review, Wynn and Eckert (2017) defined categories as micro, macro, empirical, demonstrated models, and a definition of an iterative stereotypes. For the analysis of the approaches to iteration in the analysis of the literature, this thesis, suggest the use of a taxonomy with six categories to support the understanding of the different interpretations of the design iteration. The first category is defined for the qualitative characteristics of design iteration, within this classification, design iteration is perceived as having positive or negative effects, desirable or undesirable to the design process or product (Wynn and Eckert, 2017). Ballard (2000) approach to design process, from a lean design perspective, points out that negative iteration is an important source of waste, and suggest the generation of value through positive iterations, through organizational strategies to reduce iterations with management design techniques.

Le, H. N. (2013) suggested that the positive effects of iteration include exploration of concepts to find and correct flaws, enabling development under uncertainty and change. Wynn and Eckert (2017) asserted that design iteration has both positive and negative effects, but they likely depend on situation-specific factors. However, from a strategic perspective, when the iteration is used during the design process, it is expected that appropriate strategies and policies influence the process. Yassine et al. (2003) suggested that a reduction of the pairwise coupling reduces the instability caused by iterations leading to a more rapid completion.

On a different note, Bhuiyan et al. (2004) found that increments in functional interactions decrease cross-phase iteration caused by preliminary information reducing the overall effort and time, even when more iteration is required. On a different note, an operational approach of iteration is when it is used or expected during the overlapping of tasks in concurrent strategies. For example, in Eppinger et al. (1994), the use of the concurrent iteration developing tightly coupled subsystems through frequent information exchange.

The organizational aspects of the iteration considers iterations happening due to the interactions between product, process and organisational levels (Le, H.N. et al., 2012). For instance, Eckert et al. (2014) highlighted the uncertainty in decision making related to technology, communication, and new design solutions, when design teams, customers, and suppliers iterate to converge on an effective design solution. On the other hand, Piccolo et al. (2019) attempts to connect design iteration with a social perspective, arguing that social networks influence iterations. In their study, they find that the number of iterations increase when the number and influence of stakeholders exhibits a prominent role as facilitator or authority.

A product perspective of design iteration considers iterations directly influencing the current state of design (Wynn and Eckert, 2017), within the problem and solution space (Dorst and Cross, 2001), including the enhancement of characteristics and attributes. The process approach identifies how design iteration influences the design or development processes, for instance, when iterations are the result of previous iterations (Jin and Chusilp, 2006), or designers proceed with similar patterns of design activity (Costa and Sobek, 2003).

Wynn and Eckert (2017) distinguished between three broad categories for design iteration stereotypes: (1) iterations toward progression, (2) iterations for corrections, and (3) coordination. Progressive iterations provide a better design by contributing to the problem solution, the refinement of specifications, and functionalities. In contrast, corrective iterations often respond to an unplanned event; they are perceived as

undesirable when they cause rework or new work or can produce a cascade effect when a solution to a problem causes other problems. On the other side, the coordinative iteration helps to do the process effectively and efficiently.

Table 2.5.1.1 shows the literature approaches identified in the analysis. The first column shows the author and publication year, followed by five columns with the names of the different perspectives discussed above.

Table 2.5.1.1 Design Iteration perspectives in literature based on Wynn and Eckert, 2017. Figures at the bottom of the table represent the percentage of studies that addressed each theme.

Publication	Qualitative	Strategic	Operational	Organizational	Product	Process
Asimov (1962)	✓			✓	✓	✓
Galbraith (1974)	✓					✓
Eastman (1980)			✓			
March (1984)		✓				
Clark et al. (1987)			✓	✓		
Gero (1990)						✓
Guindon (1990)					✓	✓
Hybs and Gero (1992)						✓
Schon and Wiggins (1992)	✓				✓	
Smith et al. (1992)		✓				
Smith and Eppinger (1993)	✓					
Clausing (1994)	✓			✓		
Eppinger et al. (1994)	✓		✓			
Bucciarelli (1994)	✓		✓	✓		
AitSahlia et al. (1995)			✓			
Eisenhardt and Tabrizi (1995)					✓	✓
Ha and Porteus (1995)		✓	✓			
Krishnan et al. (1995)		✓				

Ward et al. (1995)						✓
Kolodner and Wills (1996)					✓	
Maher and Poon (1996)					✓	✓
Safoutin and Smith (1996)	✓	✓		✓	✓	
Cusumano (1997)					✓	
Cusumano and Selby (1997)					✓	
Iansiti and MacCormack (1997)					✓	
Krishnan et al. (1997a)			✓			
Krishnan et al. (1997b)		✓				
Smith and Eppinger (1997a)		✓	✓			
Thomke (1997)					✓	✓
Braha and Maimon (1998)		✓			✓	✓
Browning (1998)			✓			
Loch and Terwiesch (1998)		✓				
Smith and Leong (1998)					✓	
Smith and Tjandra (1998)			✓			
Thomke (1998)						✓
Ahmadi and Wang (1999)		✓	✓			
Atman et al. (1999)					✓	
Hoedemaker et al. (1999)					✓	✓
Love et al. (1999)	✓					
Sobek et al. (1999)			✓			
Terwiesch and Loch (1999)	✓	✓				
Adams and Atman (2000)					✓	✓
Ballard (2000)	✓		✓			
Isaksson et al. (2000)	✓					
Love and Li (2000)	✓					
Roemer et al. (2000)			✓			

Austin et al. (2001)				✓	✓	✓
Dorst and Cross (2001)					✓	✓
Joglekar et al. (2001)		✓	✓			
MacCormack et al. (2001)	✓					
Love (2002)		✓				
Terwiesch et al. (2002)						✓
Ahmed et al. (2003)					✓	✓
Badke-Schaub and Gehrlicher (2003)	✓		✓		✓	✓
Costa and Sobek (2003)				✓	✓	✓
Loch et al. (2003)		✓				✓
Mihm et al. (2003)						
Safoutin (2003)	✓				✓	✓
Yassine et al. (2003)		✓		✓		
Bhuiyan et al. (2004)		✓	✓			✓
Eckert et al. (2004)					✓	✓
Love and Edwards (2004)	✓					
Cho and Eppinger (2005)		✓			✓	✓
Fairley and Willshire (2005)	✓			✓	✓	✓
Huberman and Wilkinson (2005)	✓					
Boudouh et al. (2006)	✓					✓
Chusilp and Jin (2006)	✓					✓
Jin and Chusilp (2006)				✓	✓	
Liker and Morgan (2006)					✓	
Taylor and Ford (2006)	✓				✓	✓
Braha and Bar-Yam (2007)				✓	✓	
Wynn and Eckert (2007)	✓	✓			✓	✓
Dyba and Dings0yr (2008)					✓	✓

Jun and Suh (2008)	✓		✓	✓		✓
Arundachawat et al. (2009)					✓	✓
Hatchuel and Weil (2009)					✓	✓
Hwang et al. (2009)	✓					
Jin and Benami (2010)						✓
Love et al. (2010)						✓
Moen and Norman (2010)					✓	
Le (2012)	✓					
Schlick et al. (2013)			✓			
Fernandes et al. (2014)						✓
Haller et al (2014)	✓					
Kim et al. (2014)	✓					
Eckert et al. (2014)	✓			✓		
Frillici et al. (2016)		✓				✓
Moore et al. (2016)	✓	✓		✓	✓	✓
Wynn and Eckert (2017)	✓	✓			✓	✓
Yassine et al. (2018)		✓				
Browning (2018)	✓	✓		✓	✓	✓
Picciolo (2019)				✓	✓	✓
Hassanezhad et al. (2019)		✓		✓	✓	✓
Singht et al. (2019)		✓		✓	✓	✓
TOTAL	18%	14%	11%	10%	23%	24%

The results of the analysis of the approaches to design iteration in the literature led to conclude that most of the research endeavours focused on the analysis of design iterations regarding product and processes followed by the negative or positive effects of the design iterations. The organisational aspects of design iteration are at the time of performing of this analysis, the areas where more research is required.

Table 2.5.2.1 summarizes the publications by quantity, perspective, and year, showing that design iteration has been recognized as an important feature of the design process since the early works of Asimow (1962). However, it is only during the last three decades that the analysis of the impacts of design iteration, from the engineering design research community, took a higher relevance from 1997 to 2003 with many publications. These publications were mainly looking to develop understanding about how iteration influences products and processes, followed by enquiring about how the feature contribute or impact the process or product.

Table 2.5.1.2 Quantity of publications in different perspectives between the years of 1962 to 2020.



This review found that Safoutin and Smith (1996), Costa and Sobek (2003), Wynn and Eckert (2017), and Jin and Chusilp (2006) developed a more holistic view and proposed frameworks considering not only aspects of design iteration, but also identified design iteration through different abstraction levels, domains, and actors during the design or product development process.

2.5.2. Micro, meso and macro iterations

Safoutin and Smith (1996) asserted that the micro, meso, and macro are terms used to refer to broader loops in scope, such as the successive model releases of a product the marco level, or narrower than such mathematical techniques used for an engineering optimisation process the micro level. They distinguished between three main scales for design iteration: micro, meso, and macro. The micro-scale iteration is related to a low level of design problems; it is an error-driven design process. The meso-level iteration links distinct stages of the design process through a proposal, testing, and modifications cycle. Finally, the macro scale iteration is a refinement of the design with a product annual release.

Costa and Sobek (2003) defined a framework with three iteration types. First, the rework iteration that occurs with the repetition of the activity at the same level of abstraction, in the same object, generally to correct an error. Second, the design iteration, which is an activity where the design evolves toward the desired final state. This iteration repeats the activity to generate meaningful information that helps designers define and refine a solution. Third, the behavioural iteration that proceeds through the same activity, at the same abstraction level, with different scope This means, for instance, that designers divide a problem into parts and perform a similar pattern of design activity on each part.

2.5.3 Mental iteration and iteration of the design task

Jin and Chusilp (2006) proposed a framework to study mental iteration in different design situations. The authors identified to types of design iterations, (1) the iteration of the design task as the repetition of task often carried out by design teams, due to new information arriving or because of a previous iteration, and (2) the mental iteration, characterized by the repetition of the cognitive activities of a single designer while he/she performs design tasks.

By using a IDEF0 diagram the authors modelled mental iteration loops depicting the 'flows' of design contents between cognitive design activities (Figure 2.5.3.1).

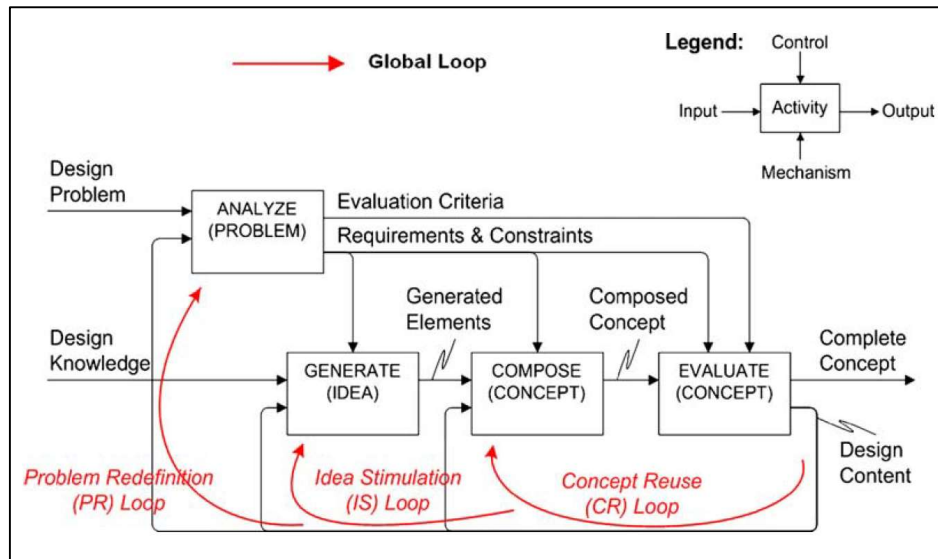


Figure 2.5.3.1 A cognitive activity model of conceptual design, from Jin and Chusilp (2006).

Jin and Chulisp (2006) argue that mental iteration may be modelled as a sequence of transitions between information processing activities and decision activities, looping within and among a number of design-specific cognitive activities. Their model comprises four key cognitive activities of the generation of the idea process, analyse, generate (idea), compose (concept) and evaluate (concept). The model allows the exploration of the relations between mental activities, identifying roles of various content, in mental iteration with respect to different phases of thinking in conceptual design (Chusilp and Jin, 2006).

2.5.4 Modelling approaches to design iteration

Several authors have developed algebraic and mathematical models of how iteration is created and when tasks are overlapped, and to study the optimal timing of design reviews in concurrent processes (Krishnan, Viswanathan et al., 1995); (Roemer et al., 2000). The mathematical models consider management and time of testing, such as the model of Ha and Porteus (1995) which studies the optimal timing of design reviews in the presence of concurrency.

Another set of studies analyse the relationship between design freeze and iterations, for example the models of Krishnan, V. et al. (1997), Keller et al. (2008), and Lee, J. and Hong (2015). In addition, they study the complexity associated with the interrelation of design tasks and design problems influencing design time. A number of authors have developed models approaching the increments in complexity with revisiting the design task (Braha and Maimon, 1998; 2013), the complexity

associated with the connectivity patterns of the tasks (Loch, C. et al., 2003), and the complexity raised from the coupling density (Yassine et al., 2003).

Other models have attempted to resolve dense cycles of information dependency and iteration (Smith, R., P. and Eppinger, 1997a; Loch, C. et al., 2003; Huberman and Wilkinson, 2005), consider how the coordination of participants may influence the iteration, during the overlapping of tasks (Loch, C.H. and Terwiesch, 1998), or analyse the decomposition of the interdependent design work and information sharing between teams (Yassine et al., 2003). Different models simulate how communication overload influence the amount of iteration (Levitt et al., 1999), and modelling series of iterations in a concurrent design tasks to select values for design parameters (Mihm et al., 2003; Loch, C. et al., 2003).

A model serves as a tool to specify and organize the understanding of a system with the purpose of explaining and communicating (Chakrabarti and Blessing, 2016b). Although modelling iteration is a relevant for the understanding of the behaviour of the design processes (Wynn, 2007) and simulation models in literature address different perspectives, there is still a lack of research on how to represent the iterative dynamics of the new product development processes in a relatively simple representation that can be manipulated visualised and validated by discussion with process participants (Wynn,2007).

2.6. Rework

Within product development, both strategy and early design decisions influence the organisational structures needed to develop engineering designs and the social networks formed by design teams. As designs develop, new information and constraints emerge, and design requirements change, leading designers to revisit and reevaluate design decisions. During these processes, iterations contribute to the quality of the design and progression (Wynn, Eckert 2017). However, iterations also increase project duration and cost, and cause rework when these iterative cycles propagate into different stages. Rework is recognized as the unnecessary effort and delays arising from redoing a process or activity not adequately implemented the first time (Love, P.E., 2002), due to initially imperfect information or changes in requirements (Smith, R., P. and Eppinger, 1997a), consuming time (Arundarachawat et al. 2009) and affecting project duration and cost.

2.6.1 Causes of rework

From the perspective of Engineering Design and product development, Costa and Sobek (2003) define rework as the repeating of an activity at the same scope and

abstraction level, while Repenning (2001) refers to rework as the unplanned allocation of resources to fix problems discovered late in the product development cycle. Wynn and Eckert (2017) defined rework as redoing tasks in similar way because the inputs and assumptions have changed, and Mitchell and Nault (2007) recognised rework as a design change whose implementation alters work that was previously done upstream and downstream. Kennedy et al. (2014) define rework as the unnecessary work because a prior decision was assumed to be final and is changed because it was found to be defective. For Taylor and Ford (2006a) rework is the task that needs to be redone because a change. Cho and Eppinger (2001) pointed out that feedback rework occurs because a downstream task fails to meet established criteria, and feed-forward rework occurs on a downstream task because new information arises from an upstream task. Dullen et al. (2019) assert that the major influence of rework during the project execution are related to the task dependency, process execution project complexity, information evolution, and information completeness.

When new information is obtained due to overlapping tasks, rework might happen because inputs change after rework on other tasks or because the outputs fail to meet established criteria. In the information evolution, uncertainty refers to a technical problem where the problem is understood but the value of its variables is unknown, and ambiguity is used to refer a situation where neither the variables nor a mechanism to solve the problem to increase knowledge is recognized (Schrader et al., 1993).

Information stability is the likelihood that the preliminary information does not change for the remainder of the process (Dullen et al., 2019). However, when ambiguous problems are known, the causes of instability in information are the evolution of information, as a result the evolution of the information brings stability and precision (Terwiesch et al., 2002). Consequently, the extent of rework will be a function of the information evolution and the downstream sensitivity (Dullen et al., 2019), which refers to the extent to which changes in the upstream information create rework in the downstream activity (Bogus et al., 2006). The faster the evolution of the upstream activity, the less likely the upstream information substantially changes (Bogus et al., 2006); hence, there are fewer changes influencing the downstream activity and less rework in consequence.

In the interdependent tasks, the downstream activity is dependent on upstream preliminary information, and the uncertainty and information instability generated during the process using preliminary information causes a high risk of reworking tasks, ultimately causing a vicious cycle of iterations that leads to the churn effect

(Dullen et al., 2019). The churn effect is described by Wynn and Eckert (2017) as the ongoing corrective iterations in which solving problems creates more problems without terminating quickly. Yassine et al. (2003) define design churn as the scenario where the total number of problems being solved (or process being made) does not reduce (increase) monolithically as the project evolves over time.

On a different note, the development of a complex system requires multiple integrated engineering design teams to develop systems, sub-systems, and components, the major number of teams working in a system, and the major risk of misalignment in the product architecture and organizational structure (Sosa et al., 2004). The misalignment leads to sub-optimal designs and interface issues identified later in the life cycle that will need rework (Dullen et al., 2019).

The definition of rework varies depending on the perspective and context. Some authors define rework as a repeating an activity at the same scope and level (Costa and Sovek, 2003), while others refer to it as the unplanned allocation of resources to fix late discovered problems (Repenning, 2001), or redoing tasks due changing inputs and assumptions Mitchell and Nault (2007). The literature analysis led to conclude that rework is the need to redo a task due a wrong, incomplete or new information, in a later or earlier stage of a product development project, and the occurrence of rework depends on the size or complexity of the product or product development project. The analysis led to conclude as well that the literature does not highlight the organisational implications on project development that impact rework.

2.6.2 Mitigation of rework

An important trend in engineering design research literature has developed through different research approaches: strategies for the mitigation of rework. The table 2.6.2.1 present a summary of the relevant models and frameworks developed by academics and practitioners. The first column is used to allocate the author's last names, in form of bibliographic reference. The second column is used to identify the methodological approach, we use worlds as model or framework or other when is appropriated. The following columns provide a brief description of the outcomes or expected behaviour; we use a verb at beginning of the paragraph to identify the aim of the work, for instance "optimize, determine, account, study" in most of the descriptions.

Table 2.6.2.1 Relevant frameworks developed for the mitigation of rework based on Dullen et.al, (2019).

Authors	Methodological Approach	Address	Estimate	Determine	Evaluates	Identify	Improve	Optimize	Reveal	Study	Understand	Use	Concept
Ha and Porteus (1995)	Dynamic model							✓					Design reviews.
Krishnan, Viswanathan et al. (1997)	Framework			✓							✓		Optimal timing for overlapping activities.
Loch and Templesch (1998)	Analytical model.												Information dependency.
Chakravarthy (2001)	Optimization model			✓									Optimal overlapping.
Roemer and Almadal (2004)	Deterministic model			✓									Optimal overlapping.
Bogus et al. (2006)	Framework					✓							Overlapping strategies.
Mitchell and Haut (2007)	Survey										✓		Impact of cooperative planning
A. Arundelachari et al. (2013)	Framework		✓										Best practices to avoid rework.
Thomke and Bell (2001)	Mathematical Model			✓									Optimal timing for sequential testing activities
Taheri et al. (2017)	Framework						✓						Overlapping testing and design activities.
Cho, S.-H. and Eppinger (2001)	DSL and ad-hoc simulation				✓								Task duration overlapping and iterations
Smith and Eppinger (1997b)	Stochastic matrix			✓									Ordering of coupled tasks
Yang et al. (2013)	Framework						✓						Team coordination.
Yassine et al. (1999)	Framework									✓			Conditions of product development that causes churn
Templesch et al. (2002)	Dynamic Model			✓									Information coordination stages
Yassine et al. (1999).	Probabilistic model										✓		Impact of task dependency/interim and cost
Jin, H.-B. et al. (2005)	Model		✓										Cycle time considering information dependency.
Nelson et al. (2016)	Network model										✓		Communication flows
Repenning (2001)	System dynamics model									✓			Causes of persistent reworks
Taylor and Ford (2008b)	System dynamics model												Features of control loops
Ford and Stelman (2003)	System Dynamics model										✓		Impact of concealing rework requirements
Yang et al. (2014) et al	Discrete events model								✓				Uncertainty related to iteration and overlapping impacts
Lefebvre and Browning (2009)	Agent-based											✓	Information to make informed decisions
Braha and Bar-Yam (2007)	Dynamic network model									✓			Engineering and Management efforts
Yassine et al. (2008)	Dynamic Model			✓									Optimal timing of information changes
Wang and Yan (2005)	Optimization model			✓									Optimal concurrency between one upstream multiple downstream process
Smith and Eppinger (1997a)	Matrix model										✓		Strongly coupled tasks
Browning and Eppinger (2002)	Discrete events simulation										✓		Impacts of process architecture, cost, duration and risk.
Sosa et al. (2004)	Network model					✓							Organizational and system boundaries impact the alignment of design interfaces
Hoedemakers et al. (1999)	Mathematical model		✓										Optimal number of modules, to minimize the expected project time completion
Yassine and Braha (2003)	Framework using DSL	✓											Fundamental problems, iteration, overlapping decomposition, and integration.

Rework is a significant component in product development cycle time, representing up to two thirds of the of the project effort according to Osborne (1993). While Kennedy et al. (2014), assert that larger companies expend about 70 to 80 percent of the development time reworking a design.

There are three major interests in the research literature about rework. In one third of the total population of research articles reviewed, the research seeks to develop understanding on how task and information dependencies, communication, cooperation, and planning impact the outcomes and timing of the development process, where rework is present. i, e., (Loch, C.H. and Terwiesch, 1998),(Mitchell and Nault, 2007), (Yassine et al., 1999), (Nelson et al., 2016), (Taylor and Ford, 2006b), (Ford and Sterman, 2003), (Smith, R., P. and Eppinger, 1997b), (Browning and Eppinger, 2002). A second third seeks to determine what are the optimal strategies, for overlapping tasks, testing activities, and concurrency, for instance, (Krishnan, Viswanathan et al., 1997), (Chakravarty, 2001), (Roemer and Ahmadi, 2004), (Thomke, S. and Bell, 2001),(Smith, R., P. and Eppinger, 1997a) (Terwiesch et al., 2002),(Yassine et al., 2008),(Wang, Z. and Yan, 2005), (Hoedemaker et al., 1999). The final third are concerned with the causes and conditions leading to rework or persistent reworks and what are the management efforts related, for instance, (Yassine et al., 2003),(Repenning, 2001), (Braha and Bar-Yam, 2007). The remaining works address, diverse strategies on how to use, optimize, or identify, or reveal, or address situations within the project development. The following graph, in figure 2.6.2.1 shows the main approaches from the reviewed literature.

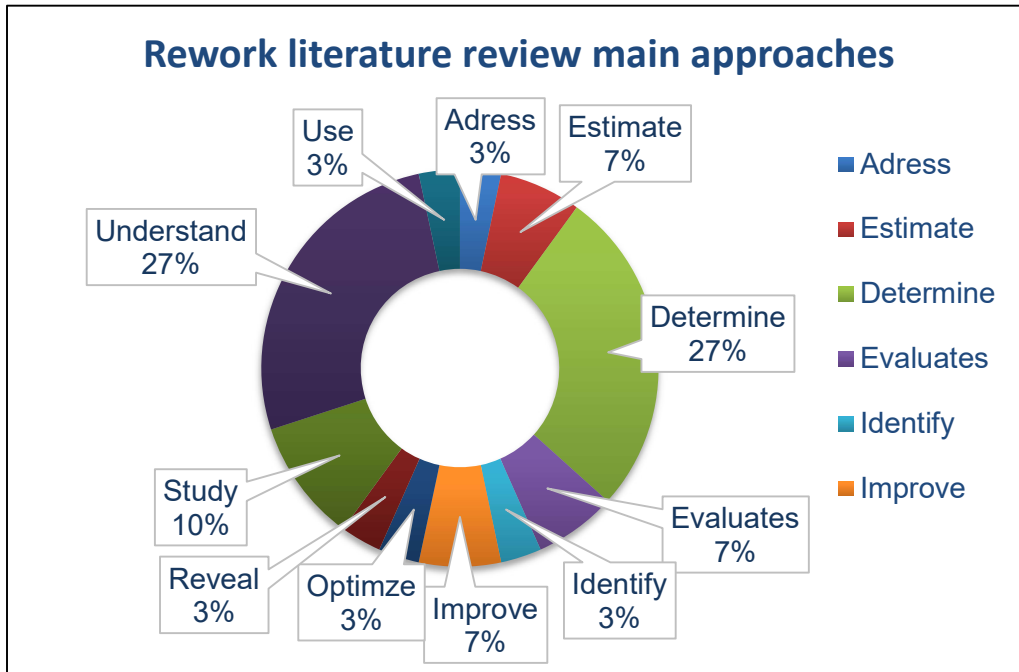


Figure 2.6.2.1 Summary main approaches on the mitigation of rework in the literature from table 2.6.2.1

The socio-technical perspective of this research considers technical, socio-aspects, and systemic connections to understand how human and organisational factors influence task performance and how technical systems are used (Clegg et al., 2017), are useful to develop understanding on task and information dependencies, communication, cooperation, and planning impact the outcomes and timing of the development process, where rework is present.

2.7. Feedback loops

Tsoukas and e Cunha 2017, states that feedback (or causal) loops are an essential concept in social and organisational theory and enhance understanding of the relationships between the past and current state systems generally used in sciences and mathematics, it is an approach to compare processes and their resulting behaviour, an essential element to understand the relationships between the past and current state of a system, that indicates the dependence of a future state of a system upon an earlier state (Tsoukas and e Cunha, 2017). The term feedback loop describes activities aimed to reduce a gap between a perceived and future state of a system. However, it is also used to suggest that the outputs of a process will influence its input directly or indirectly at some point in time. Feedback loops are also associated with the evolution of a system over time when they are related to control

and stabilisation and with the improvement or decline of a process or behaviour when they are associated with virtuous and vicious circles (Wynn and Maier 2022)

Richardson and Pugh, (1981) defined a feedback (or causal) loop as a closed sequence of causes and effects with closed paths of action and information. Conversely a linear chain of causes and effects that does not close back on itself is called an open loop (Richardson and Pugh, 1981). Monge (1990) analysed feedback loops in organizational processes from a dynamic theory perspective and suggested that feedback loops represent processes that occur over time, have a positive or negative sign, and are characterised as stable or unstable. Strand and Söderström (2002) assert that the bi-directional nature of the feedback loop between management and the core business processes allows knowledge management to develop by receiving inputs from business processes and functions to make them more effective. Kline and Rosenberg (1986) defined feedback loops as small or large recursive cycles characterising the relationships and iterations in an innovation model, in particular, among research, invention, innovation, and production. McCarthy et al. (2006), approaching new product development (NPD) as a complex adaptive system, identified overlaps in the stage-gate model which are referred to as feedback loops that facilitate the customisation of the NPD process behaviours and also configuration from linear to chaotic with corresponding types of innovation output that range from incremental to radical.

A causal loop that tends to reinforce or amplify a change is a positive, reinforcing, or deviation-amplifying feedback loop, while a closed-loop that tends to counteract a change is called a negative, deviation-counteracting feedback loop (Masuch 1985).

2.7.1 Vicious circles in organisations

Vicious and virtuous circles are prevalent in social systems, such as organisations, where there are numerous heterogeneous and often conflicting causal loops. A vicious circle is a deviation-amplifying loop that turns a challenging situation worse, and a virtuous circle is a reverse deviation-amplifying loop that makes a good situation better (Tsoukas and e Cunha, 2017). Edgeman et al. (2020) asserts that in organizations various patterns or cycles of behaviour exist, that can be identified as enterprise routines or habits that produce a range of negative to positive results.

Patterns of behaviour producing predictable poor to negative results might be regarded as vicious circles. Edgeman et al. (2020) argues that a vicious circle is a self-propagating complex chain of events with failures or negative consequences at one stage that generate increasingly serious failures or negative consequences at

each subsequent stage. In contrast, patterns or behaviour that produce predictably positive results can be regarded as virtuous circles, a self-propagating complex chain of events with positive consequences at one stage generating positive consequences at each subsequent stage (Edgeman et al., 2020).

2.8. Simulation models of product development process.

Simulation models are to Hardebolle and Boulanger (2009) models that can be executable, when there is an algorithm able to compute a behaviour accordingly to the semantics of the modelling language. Executable formalisms are modelling languages which have a formal syntax and semantics, considered unambiguously defined, even when they do not involve a mathematical definition. While a modelling executable paradigm can be considered as a mindset for modelling or a set of requirements that govern how a system is to be modelled.

An important contribution to the research is the improvement to the product development processes by using a formalism approach to model new product development processes that was made by Clarkson and Hamilton (2000), Cho and Eppinger (2005), Wynn et al. (2010), Hassannezhad et al. (2019) and Wynn and Clarkson (2021). In a different note, models using paradigm approaches were made by Ford and Sterman (1997) and Rahmandad and Hu (2010).

2.8.1 Computational models in engineering design

Nikolic and Lukszo, 2013 stated that computational models are analogies of real-world systems that inevitably involve some reduction of complexity and approximation. Their purpose is to design or represent real-world or anticipated systems such as a design concept, a facility design, or a process design. Their design and adjustments must reflect the outcomes of real-world systems, and with simulation techniques develop experiments that allow the understanding of the systems' performance under different operating conditions, to evaluate management strategies or decision-making processes, usually when prototyping or experimentation is expensive or impossible to build (Nikolic and Lukszo, 2013).

Computational modelling construction distinguishes between analytical or static models and dynamic or simulation models. The analytical model's outcome depends on the input, while dynamic modelling uses a set of rules that define how the modelled system will change in the future. Simulation is the execution that takes the model through (discrete or continuous) state changes over time (Borshchev and

Filippov, 2004). A general process of developing computational simulation models entails a series of stages starting with a (1) research question that needs to be answered, that leads (2) to establishing a definition of a target system to be modelled, (3) the collection of some observations about that target that may lead to establishing (4) the parameters and initial conditions. In the construction of the simulation model assumptions are allowed, followed by a verification stage to debug of the model and a stage for the validation to ensure that the behaviour of the model corresponds to the behaviour of the target, and a stage for experimentation that will provide with the answer to the what if? question.

Depending on the modellers' approach and simulation strategy used, those stages include activities. However, an important aspect of the construction of simulation models is related to the simplification of the model among the target system intended to be represented, while the definition of modelling establishes that a model is a simplification of the target system, it is more about a process of abstraction, that aims to reduce the space of the system by omitting details considered irrelevant (Dams and Grumberg, 2018). The outcome of the abstraction process, including aspects like problem definition, purpose, and objectives is known to be part of the conceptual model.

2.8.2 Simulation of iterative process

A task-based approach considers design iterations as the revisiting of an already completed task or the execution of similar tasks in different contexts. Task-based models tend to be mechanistic and do not account for control mechanisms in the process (Wynn, 2007). It is also recognized that Iteration is a social coordination process where actors negotiate trade-offs. Here, iteration is modelled as a function of continuous dialogue between the participants. Finally, in the information-based modelling approach, the process information determines process behaviour; here the process model aims to capture iterations feedback by releasing preliminary information prior to its final task completion (Wynn et al., 2007).

2.8.3 Simulation of engineering design teamwork

Teamwork is not a result of the simple aggregation of individual talents. In organizations, the knowledge, the flows, and the information processing of individuals (actors) in product development teams impacts many aspects of the organizational dynamics (i.e., beliefs and norms) but also the process of decision making, learning, and innovation. A common practice in the new product development organizations is the creation of multi-teams environment, where

individuals have multiple memberships and the team boundaries are often ill-defined (Crowder et al., 2012).

Agent-based modelling is recognised as an efficient tool for modelling complex socio-technical systems, which include organizations and human behaviour. The agents in these systems represent human individuals and can represent some human characteristics like motivation, memory, and learning. Agent-based models are suitable for team profiling and examination of the effect of special features on team performance (Perišić et al., 2016). Its characteristics allow the analysis and prediction of team performance, taking into account task, human, and organizational factors. Models in the literature that provide support in design problems or as environments enabling cooperation are SHARE (Toye et al., 1994), PACT (Cutkosky et al., 1993), Hao model (2006), Wang model (2009), and Madhusudan model (2005).

A methodology with agents representing specialists in a team working in the same activity is A-design developed by Campbell et al. (1999). To simulate design team behaviour, NASA developed a simulator called Team X, and lastly McComb et al. (2015) developed a modelling framework considering a theory-based characteristics of teamwork in design. Another set of models that considers work distribution, and possible problems in activity performance, is formed by VDT, (Jin and Levitt, 1996), the NetWatch (Tsvetovat and Carley, 2004), the models of Zhang et al. (2009); (2012), TEAKS of Martínez-Miranda et al. (2006), the model of Crowder et al. (2012), the model of Dehkordi et al. (2012), and the model of Singh, V. et al. (2013).

2.8.4 Simulation of the rework

System dynamic studies in product development traditionally focused on the project dynamics, including project evolution, with different complexity levels and capturing different feedback effects. The models representing these trends are Cooper, K.G. (1980), Abdel-Hamid and Madnick (1991), Taylor and Ford (2006b), and Lee, S. and Peña-Mora (2007). The rework cycle can be recognized by the need for rework due a flawed project task. The cycle can repeat itself, extending the project duration far beyond the project original duration. However, in the absence of this, the project completion is a function of the number and scope of the tasks, the available resources, and productivity. The rework cycles generate a path dependent reinforcing loop, which considers defects, quality and testing, and the study of this is central to understand the project delays and disruptions.

2.9 Multiparadigm simulation modelling

The multiparadigm modelling approach, has not been an exclusive concern of the models and simulation domain. Process models have taken an increasing importance for the development process in systems engineering. A significant proliferation of dedicated modelling languages intended to capture specific knowledge, adapted know-how, and contributions to the efficiency, productivity and quality of the systems is now available (Hardebolle and Boulanger, 2009). The Multi-Paradigm Modelling (MPM) method consolidates different modelling methods and techniques, enabling engineers to model each aspect of the system explicitly at the most appropriate abstraction level (Challenger et al., 2020).

In the modelling and simulation domain, a traditional modelling and simulation stand-alone approach faces serious challenges to represent the overall multidimensional nature of a system like product development process (Mykoniatis and Angelopoulou, 2020). The multi-paradigm simulation approach allows the generation of interoperable simulations able to capture interactions, among elements of different abstraction levels, to address a larger range of modelling questions with a reduced amount of computational effort (Mykoniatis and Angelopoulou, 2020). The Djanatliev and German (2013) frameworks for multi-paradigm simulation models suggests three significant processes to structure the simulation scope: First, independent levels of abstraction or views on the system. Second, the explanation of how simulation models are linked to the abstraction levels and how the simulation paradigm is used to model structures at the considered level. Finally, to identify the connections that reflect the interactions between abstraction levels.

The literature related to simulation modelling identifies three modelling methods or approaches available to represent the real-world systems in modelling and simulation. System dynamics, where a high abstraction level is used, thinks in terms of aggregates (stocks and flows) and feedback loops. Discrete events modelling is a process-oriented approach, where the system is represented as a sequence of operations performed over activities. The agent-based modelling approach, where individual objects interact with each other and with the environment (Borshchev, 2013). Mykoniatis and Angelopoulou (2020) provide a framework integrating those different simulation approaches in different domains such as socio-technical, cyber-physical systems, business, and healthcare organizations, arguing that the combination of different simulation methods require an alignment among the problem, or system (“what”), the purpose (“why”), and the methodology (“how”), and derived three main questions for its framework: (1) Why and when does a real-world

system require multi-paradigm modelling and simulation? (2) What are the interaction points among the different simulation models used? (3) How do the simulation models interact with each other to exchange information?

Nikolic and Lukszo, 2013, state that agent-based models are constructed from a bottom-up perspective to discover possible emergent properties. These models do not have a desired state or task to be achieved but instead describe entities and allow observation of how they interact to explore possible system states. Agent-based simulation modelling does not focus on specific system components or subsystems but instead seeks to capture the behaviour of different actors in decision making, whether competing, cooperating, or negotiating. The concept of agent-based systems, which are composed of multiple interacting actors and physical elements, can simulate how system behaviour emerges from the behaviour of actors at the bottom level (Nikolic and Lukszo, 2013).

2.10 Conceptual models

In the literature related to simulation and modelling, it is still difficult to find a unanimous definition of conceptual model (Fujimoto et al., 2017). Wilsdorf et al. (2020) suggested that approaches regarding the content and specifications to conceptual modelling have a narrow or wider scope and can be seen as a formal or informal constructs. Specifically, a narrow view defines a conceptual model as an abstract description, while a wider view refers to a loosely coupled construct integrating different artefacts. Even broader, when integrating features of the model context, the formal and informal aspects are related to specifying the conceptual model and its parts, ranging from the informal using verbal narratives and sketches, to the formal, by using conceptual modelling languages. Early conceptual model definitions considered this as a vague and ambiguous informal representation of modellers' thoughts. However, later interpretations emphasize the use of formal languages easy to transform into computerized models but limited to specific types of problems.

Balci (2012) defines a conceptual model as a repository of high-level conceptual constructs and knowledge specified in a variety of communication forms, intended to assist in the design of any type of large-scale complex modelling and simulation applications, keeping separated the conceptual constructs related with the model itself and the artefacts referred to the context of the simulation, for instance problem formulation, objectives, and requirements. On the contrary, Robinson (2008a; 2008b) include in the conceptual model research questions and requirements and suggest

content should be explicit like model inputs and outputs, used data, scope, level of detail assumptions and simplifications, including entities, activities, modelling approaches, and justifications. In the definition approach, by Fujimoto et al. (2017), the conceptual model is a collection of early-stage products, integrating and providing information and requirements, and developing a more explicit conceptual model, based in domain-specific languages and ontologies.

Nikolic and Ghorbani's (2011) methodological approach, for the development of simulations of complex sociotechnical systems, identifies the need for a systematic approach to conceptual model development as part of an ongoing process for standardising modelling practice. Their first two stages, system analysis and model design, include establishing the purpose of the models and identifying the problem being simulated, key stakeholders, and the system to be conceptualised.

2.11 Agile Hybrid Methodologies

The agile methodology emerged in the software development industries during the 1990s and gained significant popularity after the release of the agile Manifesto (Beck et al., 2001). Drawing inspiration from the value maximization and waste reduction culture originally established in lean manufacturing, agile values include collaboration, team empowerment, iterative and incremental development, heightened customer engagement, and adaptability to change (de Borba et al., 2019). The potential of agile methodologies to accelerate product development and create products more likely to meet customer needs is widely reported. For example, an empirical study by Serrador and Pinto (2015) highlighted that the utilization of agile methods significantly impacted project success, particularly in terms of efficiency and stakeholder satisfaction.

Reiff and Schlegel (2022) assert that hybrid project management, which combines both traditional and agile project management techniques, capitalizes on the strengths of each approach. However, the multitude of hybrid methodologies makes it challenging to distinguish differences, similarities, advantages, or disadvantages (Reiff and Schlegel, 2022). Furthermore, there is fragmented knowledge regarding prerequisites and success factors for the successful implementation of hybrid project management within organizations (Reiff and Schlegel, 2022).

Stelzmann (2012) proposed a classification scheme to define the context for agile, considering feasibility (where rapid prototyping, testing, and implementing changes are feasible, and the system is not compromised in terms of safety) and demand for agile (where high market dynamism, innovation, and rate of change are necessary).

Because the history of many agile approaches occurs in small, highly collaborative environments, they work very well within those spaces. However, agile methods struggle when those environments face challenges (Ahmed, T. et al., 2014).

Several factors make agile methods difficult to apply in practice. Ahmed et al. (2014) pointed out that agile methods are challenging to implement within larger teams, particularly when the teams are geographically distributed and dealing with complex systems. They also note that agile methods face difficulties in projects with audit, regulatory, or safety-critical requirements, or expected higher quality requirements, as well as in projects with strict contractual commitments involving complex user environments or when the end user is not available. Agile methodologies also struggle when subcontracted into a project being run in a non-agile way (Ahmed, T. et al., 2014).

In addition, traditional project management approaches and agile project management fundamentally differ in their structures and processes. For example, the traditional approach involves determining project scope, time, and cost early in the life cycle and carefully managing scope changes (Fernandes, G. et al., 2018). In contrast, Bogdanova et al. (2020) assert that agile project management prioritizes flexibility in the face of changes in the environment and scope of services, focusing on functional requirements and employing short, sequential planning and execution cycles for more autonomous project teams, client feedback, and flexibility in project scope (Bogdanova et al., 2020).

While hybrid models offer a promising route for integrating agile methods into physical product development, they are still in their nascent stages and lack unanimous acceptance (de Borba et al., 2019). Thus methods have proven successful in software development, their effectiveness in other disciplines is still being determined (Kennedy & Umphress, 2011; Mosher, Kolozs, Colegrove, & Wilder, 2018; Rigby, Sutherland, & Takeuchi, 2016). The consensus is that a pure Agile approach, as implemented in software companies, does not seamlessly fit physical products, requiring some degree of adaptation (Reiff and Schlegel, 2022).

2.11.1 Scrum

The most common agile method is called Scrum (Dingsøyr et al., 2012). Scrum emphasizes incremental feature delivery and is designed to be flexible, allowing customers to change their minds during development without disturbing the ongoing effort (Sutherland & Schwaber, 2007). Unlike many agile methods, scrum has a foundation in theory, specifically complex systems science (Sutherland & Schwaber,

2007; Szalvay, 2006). Existing agile systems engineering approaches heavily favour scrum (Douglass, 2015; Dove & LaBarge, 2014; Kennedy & Umphress, 2011).

In scrum, iterations, incremental development, self-managed teams, and flexibility in the face of changing requirements are common aspects (Ziółkowski and Deręgowski, 2014). The term "iterative approach" of scrum refers to the division of the project duration into iterations or sprints, where the overall project is divided into several smaller projects (Vinekar et al., 2006). The team determines features for development, works on them, and reviews them with the customer at the end of the sprint. This close customer engagement allows for adjustments to the project's course and scope throughout its duration (Ziółkowski and Deręgowski, 2014).

Sprints are short repeating blocks of time in which key parts of the project are completed. Sprints are usually two to four weeks long (Sommer et al., 2015). Each sprint is based on a sprint backlog, which describes a set of priority features (or product increments) to be developed in the current sprint, selected because they are high priority and can be completed within the specified timeframe of the sprint. The sprint backlog outlines priority features for each sprint, and scrum's adaptability enables responses to constantly change requirements, market conditions, and project dynamics (Sommer et al., 2015). This flexibility allows adjustments without renegotiating contracts, ensuring continuous alignment between project scope and evolving needs (Reiff and Schlegel, 2022).

2.11.2 Hybrid Approaches

Reiff and Schlegel (2022) conducted a systematic literature review identifying two different streams in hybrid project management. The first combines an agile approach at the operational level and a traditional approach at the decision-making level, attempting to combine the advantages of both management systems (Binder et al., 2014). The second involves hybrid project management integrating an agile approach into a traditional project management methodology (Reiff and Schlegel, 2022). The literature review also provided the identification of four different hybrid methodologies that systematically combine traditional and agile project management phases, summarized in the following table.

Table 2.11.1 Hybrid traditional and agile project management approaches from Reiff and Schlegel (2022).

Approach	Phases		
	Initial phase	Development phase	Final phase
Water-scrum-fall	Waterfall	Scrum	Waterfall
		Design	Integration
		Development	Testing
		Implementation	
Water-Agile	Waterfall	Agile approach	Agile approach
	Requirement analysis	Design	Testing
	Planning	Development	
		Implementation	
Hybird V-model	V-model	Scrum	V-model
	User requirements	Design	Testing
	User requirements	Implementation	System testing
	Planning	Unit testing	
Agile-Stage-Gate (Scrum-stage-Gate)	Stage-Gate for administrative and strategic activities	Stage-Gate for administrative and strategic activities	Stage-Gate for administrative and strategic activities
	Scrum for operative activities	Scrum for operative activities	Scrum for operative activities
	Discovery		Testing
	Idea generation	Development	Validation
	Scoping	Implementation	Launch

The Water-Scrum-Fall methodology combines the traditional Waterfall methodology with agile Scrum. It is based on the idea that there must be a structural framework for a project, which is provided by the established Waterfall project structure. Within this traditional process approach, agile phases are integrated (West et al., 2011).

The Waterfall methodology is the best-known and simplest process model of traditional project management, operating sequentially. Phases are completed one by one, moving the product design to the end. In this methodology, each phase must be completed before moving on to the next. It is also possible to return to a previously completed phase if adjustments or corrections are required (West et al., 2011).

In the Waterfall-Agile model, the project plan is scoped, and the first agile sprint is planned before the project begins, requiring a complete project plan. However, specific details of each sprint are not defined until the first sprint takes place and is complete (Hassani et al., 2018). The stages of design and implementation are based on agile methodologies. In each iteration, requirements are defined, and customer feedback is observed. In this methodology, individual project phases are selected and assigned before the project starts but can be exchanged during the project development according to the specified amount of effort (Reiff and Schlegel, 2022).

Hayata and Han (2011) proposed the hybrid V model, where the idea is to conduct the phases with a higher abstraction level according to the V-model, while the more detailed phases are performed according to the Scrum method. The suitability of the Scrum method is due to intensive communication within the development team, supporting the implementation phase through joint iterative thinking (Hayata and Han, 2011). In the hybrid V-model, the traditional approach is applied to the project in the initial and final phases, where there is a greater need for planning. The agile approach is then applied to the development, implementation, and testing phases where the need for agility is greater (Hayata and Han, 2011)

The hybrid Agile-Stage-Gate methodology integrates agile sprints by breaking the development process into stages composed of short increments driven by short-term, minimal planning (Cooper, R.G., 2016). This adds flexibility and speed while retaining the structure of the Stage-Gate model. The Stage-Gate model provides focus, structure, and control, combining the benefits of the agile approach (Cooper, R.G. and Sommer, 2018). Each stage is composed of a series of time-boxed sprints, incorporating iterative development cycles.

The project team determines realistic goals for the sprint and then maps out an action plan to accomplish those goals (Cooper, R.G., 2017). Each day of the sprint begins with a daily scrum, or stand-up meeting, during which the team members review what was accomplished the previous day, discuss the plan for the day, and address any problems that have arisen. At each stage, the adoption of agile sprints helps to increase responsiveness and adaptability while minimizing drawbacks (Zasa et al., 2020). Its core element is a continually evolving product definition that emerges through short-term, dynamic planning (Cooper, R.G. and Sommer, 2018)

Despite the advantages of the agile methodology, it has faced increasing criticism in recent years. To expedite the development process, the agile approach focuses more on the final product than on design and documentation (Edwards et al., 2019; Alves et al., 2019), leading to the neglect of project documentation, as the development of the solution can be time-consuming and project documentation is often given lower priority (Czechowski, 2019).

Another disadvantage is the inaccuracy in time planning and budget scheduling, attributed to the constant re-prioritization of tasks (Bogdanova et al., 2020). In an organizational context, numerous barriers and challenges inhibit the realization of the agile benefits (Durbin and Niederman, 2021; Nuottila et al., 2016). In large-scale transformation projects, basic agile principles, such as team autonomy, are challenging to maintain (Gustavsson et al., 2022).

2.12 Conclusions

The literature that describes the characteristics of product development systems is scarce; the only two sources that describe the characteristics of product development systems are the contributions of Pessôa and Trabasso (2017) and de Weerd-Nederhof (1997). de Weerd-Nederhof, (1997) identified an intra and different organisational context for the new product development (NPD) systems. The extra-organisational context consists of the group of entities outside the boundary of the new products development system influencing the system without affecting its functions (e, g., other organisations, like competitors, supply chain partners or government entities such as regulatory authorities). The intra-organizational context is integrated with other business functions (such as marketing, sales, procurement, engineering, and manufacturing), potentially affecting all or part of the NPD system.

De Weerd-Netherhof (1997) defines product development organisations as “A purposeful system of people and resources which, using multiple technologies, together perform certain ‘activities’ or ‘processes’ to transform inputs into outputs”. They assert, “If the organisation’s outputs are considered useful by the environment, the latter is prepared to pay for them, which enables the organisation to fulfil its goals”.

On the other hand, Pessôa and Trabasso (2017) point out that the product development system can be understood “as a network with multiple dimensional and highly interconnected processes, where feedback loops cross these multiple hierarchical levels”. And argue that the purpose of the product development system is performing the product development processes. Product development processes are people-based, complex, and non-linear processes (Pessôa and Trabasso, 2017) that can be understood as “ information transformation processes, where the inputs consist of partially new information, and the projects and activities outputs are expected to deliver, a proven product/process design, a working prototype or a written product introduction plan” (de Weerd-Nederhof, 1997).

Ulrich and Eppinger (2012) define the product development process as “a sequence of steps and activities that an enterprise employs to conceive, design and commercialise a product”. The traditional product development processes include six stages, starting with a planning stage, then conceptual development, system-level design, detail design, testing and refinement, and finalising the product ramp-up stage. However, it is widely agreed that organisations do not follow or define a

precise and structured product development process (Ulrich and Eppinger, 2012), as each organisation may define their own development process or will use a slightly different version of another organisation.

Although design iteration is one of the most important characteristics of engineering design processes (Maier, A.M. and Störrle, 2011), there is not a coherent body of literature on the subject. A consensus model or terminology for describing design iteration or iterative situations remains elusive (Wynn and Eckert, 2017). However, from the engineering design practice perspective, Meboldt et al. (2013) differentiate “two fundamental types of iteration (1) In-stage iterations (there are iterations within a stage which do not impact on previous gate decisions); and (2) cross-gate iterations (iterations which affect decisions from previous gates and have an impact on investments and market launch)”.

The definition of rework varies depending on the perspective and context. Some researchers define rework as repeating an activity at the same scope and level. In contrast, others refer to it as the unplanned allocation of resources to fix late-discovered problems or redoing tasks due to changing inputs and assumptions. Design changes that affect previous upstream and downstream work and unnecessary work due to defective decisions are also considered rework. Feedback rework occurs when downstream tasks fail to meet criteria, while feed-forward rework occurs when new information arises from upstream tasks.

The research community that investigated design iterations and rework have developed management tools, strategies, models, and frameworks, to benefit from design iterations and rework and to prevent or minimise their negative effects. However, societal and technological changes continuously reshape organisations, product requirements and constraints. New challenges arise from the more open, complex, dynamic and networked organisations (McChrystal et al., 2015). The design of complex products will not be exempt from design iterations, several tasks will need to be repeated or outcomes updated or upgraded due to new information arising from other tasks or organisational or social factors.

Wynn and Maier (2022) argue that feedback loops describe activities aimed to reduce a gap between a perceived and future state of a system. It is also used to suggest that the outputs of a process will influence its input directly or indirectly at some point in time. Feedback loops are associated with the evolution of a system over time when they are related to control and stabilisation and with the improvement or decline of a process or behaviour when they are associated with virtuous and vicious circles (Wynn and Maier 2022)

On a different note, the literature analysis regarding the simulation of engineering design teams leads to the conclusion that most simulation models of engineering teams activities tend to neglect social processes when they aim to estimate technical performance. Also, models that focus on exploring social processes sometimes ignore project details and task performance (Perišić et al., 2018). A model able to study and measure both intangible, individual and team-level aspects and tangible aspects like time and cost could provide a more comprehensive view of the team's performance (Škec et al., 2017)

The literature analysis concludes that the product development process is a sequential set of development activities where engineering teams carry out design work, separated by stage gates where go/no go decisions are made. Together, these form the process workflow (Shepherd and Ahmed, 2000; Montagna and Cantamessa, 2017).

Meboldt et al. (2013) suggest that two types of iteration are present during the stage-gated development process: in-stage iterations, which do not impact decisions made in previous stages, and cross-gate iterations, where decisions affect decisions made in the last gates, so moving project time and cost (Meboldt et al., 2013). Design iteration is an in-stage iteration which occurs within each development stage, improving the design quality within a design stage. These iterative cycles led designers to revisit and re-evaluate previous design decisions (Wynn and Eckert, 2017), resulting in new activities and feedback loops.

Rework is a form of cross-gate iteration, affecting performance decisions made in previous stages. Rework results from information dynamics in new product development processes and is caused by the inadequacy of information due to changes in requirements, poor decisions, defective outputs or changes in implementation that alter work previously done (Smith, R., P. and Eppinger, 1997a). The engineering design processes within product development can be regarded as complex information-processing activities consisting of creating, transferring, or disseminating information (King, 1994) directed by the decisions made by individuals in design teams (Wallace and Ahmed, 2003). Engineering designers and stakeholders interact during design activity to find a design solution (Montagna and Cantamessa, 2017). Communications, negotiation and coordination mechanisms determine the outcome and progress of the design work (Hoegl and Weinkauff, 2005; Maier, A.M. et al., 2007). The product architecture also influences technical communications and interactions among design teams (Clarkson and Eckert, 2010)

Chapter 3: Research methodology

This chapter introduces the overall research process used in developing the conceptual framework using Sein et al.'s (2011) Action Research Method (ADR). Section 3.1 summarises the method's main stages and task principles. Section 3.2 details how the Action Design Research method was used to develop the conceptual framework. Section 3.3 introduces details used in the bicycle design case study. A summary of the chapter and its relationship to the remainder is provided in Section 3.4.

This thesis has established that two kinds of feedback loops are relevant features of product development processes. Rework, governed by the stage-gated processes, is well-suited to discrete event simulation. Moreover, design iteration is driven by individual designers and so is best modelled using agent-based simulation. The integration of those simulation methods suggests that a multi-paradigm simulation approach was needed. The proposed framework by Mykoniatis and Angelopoulou (2020) for the development of simulations integrates agent-based, discrete event, and systems dynamics simulation approaches, including phases of (1) conceptual modelling, (2) simulation model development, (3) verification and validation, and (4) results and documentation. It was used to provide a suitable structure for integrating the two required simulation methods. It includes three relevant questions: (1) Why and when does a real-world system require multi-paradigm modelling and simulation? (2) What are the interaction points among the different simulation models used? (3) How do the simulation models interact with each other to exchange information?

The first question of Mykoniatis and Angelopoulou's (2020) framework was answered by identifying the key characteristics of design iteration and rework that require a multi-paradigm approach. The conceptual models reported in Chapter 4 take account of, and begin to answer, the last two questions.

Within Mykoniatis and Angelopoulou's (2020) first stage (conceptual modelling), Nikolic and Ghorbani's (2011) methodological approach for developing simulations of complex socio-technical systems was used. Nikolic and Ghorbani's (2011) stages are typical in software engineering design methodologies but include several iterative sub-steps specific to simulation modelling. Their first two stages, System Analysis and Model Design, include establishing the purpose of the models and identifying the problem being simulated, key stakeholders, and the system to be conceptualised. Next, in the model design stage, agents and interactions between them are

identified. The first two stages map onto Mykoniatis and Angelopoulou's (2020) Phase 1, and the final three stages map onto phases 2-4.

3.1 Action design research (ADR)

Action Design Research (ADR) is a method for generating prescriptive design knowledge through building and evaluating ensemble IT artefacts in an organisational setting (Sein et al., 2011). Ensemble artefacts are defined explicitly as the material and organisational features that are socially recognised bundles of hardware and software (Orlikowski and Iacono, 2001). This definition reflects a “technology as structure” view of the ensembled artifact, where structures of the organisational domain are inscribed into the artifact during its development and use (Orlikowski and Iacono, 2001). In ADR, artifact development is an iterative process, presenting a researcher-driven alpha version of the artifact, refined after successive iterated versions. Practitioners' feedback contributes to results in one or more beta versions of the artifact, in this study, the conceptual framework, where the value and utility of the outcomes are assessed. (Pettersson and Lundberg, 2016).

The ADR method includes four stages, (1) Problem formulation, (2) Building, intervention, and evaluation, (3) Reflection and learning, and (4) Formalisation of learning (Cronholm and Göbel, 2022). In ADR, each stage involves principles and tasks (shown in figure 3.1.1.1), which are described in Section 3.1.1

3.1.1 The four stages of ADR

This section outlines the stages and principles for performing Action Design Research based on Sein et al. (2011) and Lüftenegger (2020). Each stage is guided by principles and implies the performing of tasks.

ADR Stage 1: Problem formulation. Two principles drive this stage: Practice-inspired research and Theory ingrained artifact. The former emphasises viewing problems as knowledge-creation opportunities. While the latter emphasises that theories inform artifacts created and evaluated within ADR.

ADR Stage 2: Building, Intervention, and Evaluation (BIE). Activities in this stage use the problem framing and theoretical premises adopted in stage one. These stages provide a platform for generating the initial design of the IT artifact (Sein et al., 2011). Three principles drive this stage: Principle 3, reciprocal shaping, states that an ADR team formed by academics and practitioners engages in an iterative artifact development process. Principle 4, mutually influential roles, stresses the importance of mutual learning from the participants within the ADR process. Finally, Principle 5, authentic and concurrent evaluation, emphasises that evaluation is not a separate stage of the research process that follows building, and instead is related to designing, shaping, and reshaping the ensembled artifact and intervening in

organisational work practices. However, their specific format may vary based on the BIE form (Sein et al., 2011).

ADR Stage 3: Reflection and learning. In this stage, reflection on the development process, from building a particular solution to a broader class of problems, is conducted. The resulting artifact, also referred to as "the ensemble", will reflect the original design and the practitioner's perspectives within the organisational use. This stage works in parallel with Stages 1 and 2.

ADR Stage 4: Formalisation of learning. In this stage, the formalisation of the outcomes results in a tool for solving a class of problems. Principle 7, Generalised outcomes drive this stage: The resulting ensemble is, by definition, a bundle of properties in different domains. The ensemble represents a solution that addresses a problem (Sein et al., 2011).

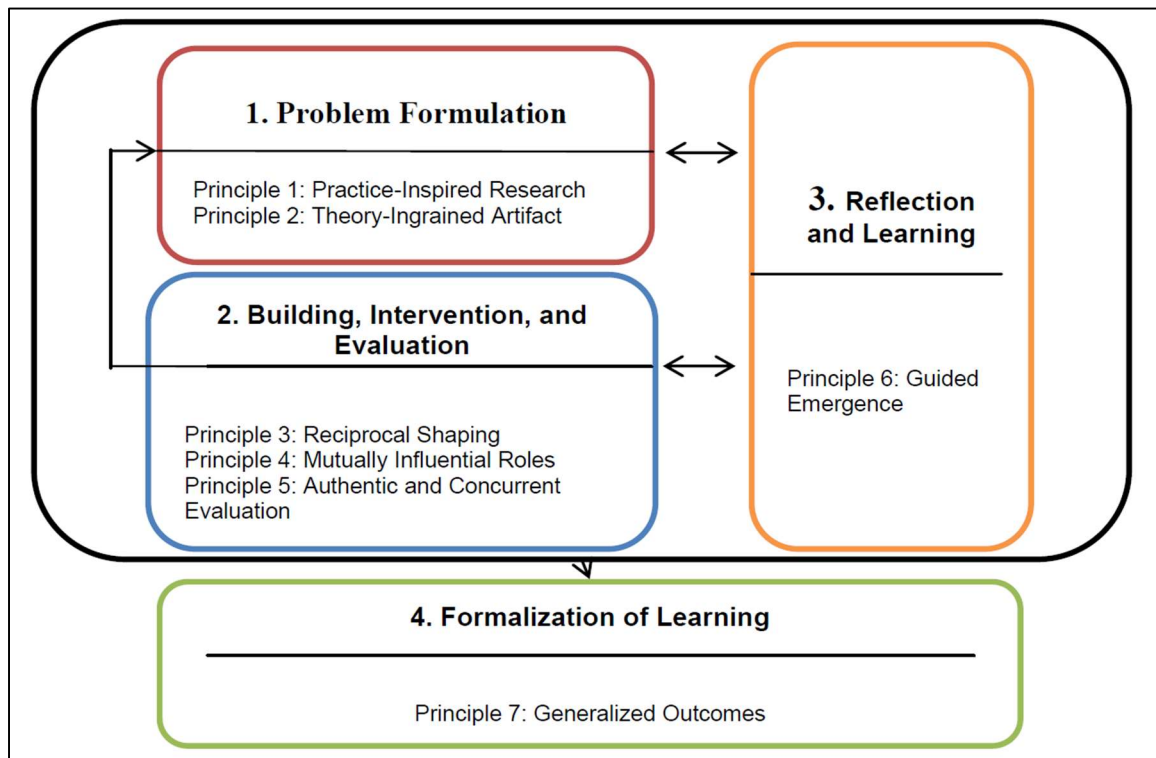


Figure 3.1.1.1 ADR Method, stages and principles from (Sein et al., 2011).

In ADR, Sein et al. (2011) suggest the tasks shown in Table 3.1.1.1 for each stage described in the following sections.

Table 3.1.1.1 Tasks in each stage, adapted from Cronholm and Göbel (2022).

<p>Stage 1. Problem formulation.</p> <ol style="list-style-type: none"> 1) Identify and conceptualise the research opportunity. 2) Formulate initial research questions 3) Cast the problem as an instance of a class of problems 4) Identify contributing theoretical bases and prior technology advances 5) Secure long-term organisational commitment 6) Set up roles and responsibilities. <p>Stage 2. Building, Intervention and Evaluation.</p> <ol style="list-style-type: none"> 1) Discover the initial knowledge-creation target 2) Select or customise the BIE form 3) Execute the BIE cycle(s) 4) Assess the need for additional cycles, repeat <p>Stage 3. Reflection and Learning.</p> <ol style="list-style-type: none"> 1) Reflect on the design and redesign during the project 2) Evaluate adherence to principles 3) Analyse intervention results according to stated goals <p>Stage 4. Formalisation of learning.</p> <ol style="list-style-type: none"> 1) Abstract the learning into concepts for a class field problem 2) Share outcomes and assessments with practitioners 3) Articulate outcomes in light of theories selected 4) Formalise results for dissemination

3.2. ADR in this research

The ADR method provides the necessary macro-structure (Cronholm and Göbel, 2022) for building and evaluating ensemble IT artifacts in an organisational setting (Sein et al., 2011). At the same time, it allows the integration of methods that support the micro-level for the construction of the socio-technical resources in the form of models, methods, algorithms or digital tools and creates design knowledge (design principles) (Cronholm and Göbel, 2022).

The current definition of ensemble IT artifacts, where structures of the organisational domain are inscribed into the artifact during its development and use, matches the socio-technical approach used in this research. Specifically, socio-technical analysis

considers technical, socio-aspects, and systemic connections to understand how human and organisational factors influence task performance and how technical systems are used (Clegg et al., 2017). Furthermore, the iterative nature of the ADR methodology and successive design constructs shape the interpretation and understanding of the organisational environment (Sein et al., 2011).

3.2.1 ADR Stage: Problem formulation

The input for this stage can come from practitioners, end users, researchers, technologies, and reviews of previous research. It is often coupled with an empirical investigation (Sein et al., 2011).

In the problem formulation stage, the research opportunity was detected through the literature analysis identifying the main characterisations of feedback loops in the context of the engineering design processes within the product development systems. Sein et al. (2011) suggest that the research opportunity should be identified at the intersection of technological and organisational domains.

3.2.1.1: Conceptualise the research opportunity.

A product development system can be seen as a multidimensional network of highly interconnected processes with feedback loops crossing multiple hierarchical levels. It can be understood as a complex socio-technical system with three relevant system elements: people, process, and product. The conceptualisation of the research opportunity consists of the understanding that realistic representations of the product development process reflect organisational and process characteristics that mirror the product architecture, which ultimately influences the process and organisational structure of the development activity. The research opportunity detected suggests the need for a conceptual framework for simulating feedback loops in engineering design that allows the understanding of the system's performance through the experimentation under different operating conditions and the evaluation of management strategies or decision-making processes (Hughes et al., 2012). The integration of two simulation methods supports the framework. It captures the social and behavioural patterns of the actors, the effects on the progress of design work and the process, the logical and chronological structure of the project and the product architecture defining the system's boundaries, and the different technical configurations.

3.2.1.2 Formulate initial research questions.

The reflection behind the identification of the research question comes from the understanding that feedback loops are critical characteristics of engineering design

processes that increase complexity, time to market and cost. Feedback loops positively impact the design outcomes (i.e., quality of final design) (Tapia et al., 2021); however, loops resulting from rework have a positive impact on the final design, but their detrimental impact on the project development outcomes may be high.

In design processes, feedback loops within the stages, known as in-stage iterations, are recognised as learning cycles essential to find the best solutions (Meboldt et al., 2013). On the other hand, cross-gate iterations are feedback learning cycles too, but when functional issues appear in the process, these feedback loops may have an impact in other stages. From the socio-technical systems perspective, positive feedback loops are virtuous circles that improve a situation. On the contrary, negative feedback loops are vicious circles that worsen a challenging situation (Tsoukas and e Cunha, 2017; Masuch, 1985).

The initial research questions were:

- (1) How do iteration feedback loops influence the product development processes?
- (2) To what extent do the process, the product, and the people interactions influence product development?
- (3) What are the main system characteristics representing those system elements?

3.2.1.3 Cast the problem as an instance of a class problem.

The identified class of problems emerged from an analysis of the design process that involved identifying the main elements of the engineering design process where iteration feedback loops emerge. The chronological structure (the process) combined with the corresponding product architecture (the product), delineates specific in-stage feedback loop iterations of individual designers and design teams (the people) and cross-gate feedback loop iterations in rework that influence the product development performance and outcomes.

Figure 3.2.1.1 depicts the coupling of the product development process with the product architecture and the in-stage and cross-gate iterations. Review gates separate the stages of the development process. The product structure is identified as “product A” in the systems level stage and “PA” in the detailed stage, “product B” in the systems level stage as well and “PB” in the detailed stage and “product C” in a similar way in the systems level stage and “PC” in a detailed stage. In-stage iterations between designers are green arrows in the detail design stage. Rework cross-gate iteration between the detail design and testing and refinement stage is identified with the yellow arrow.

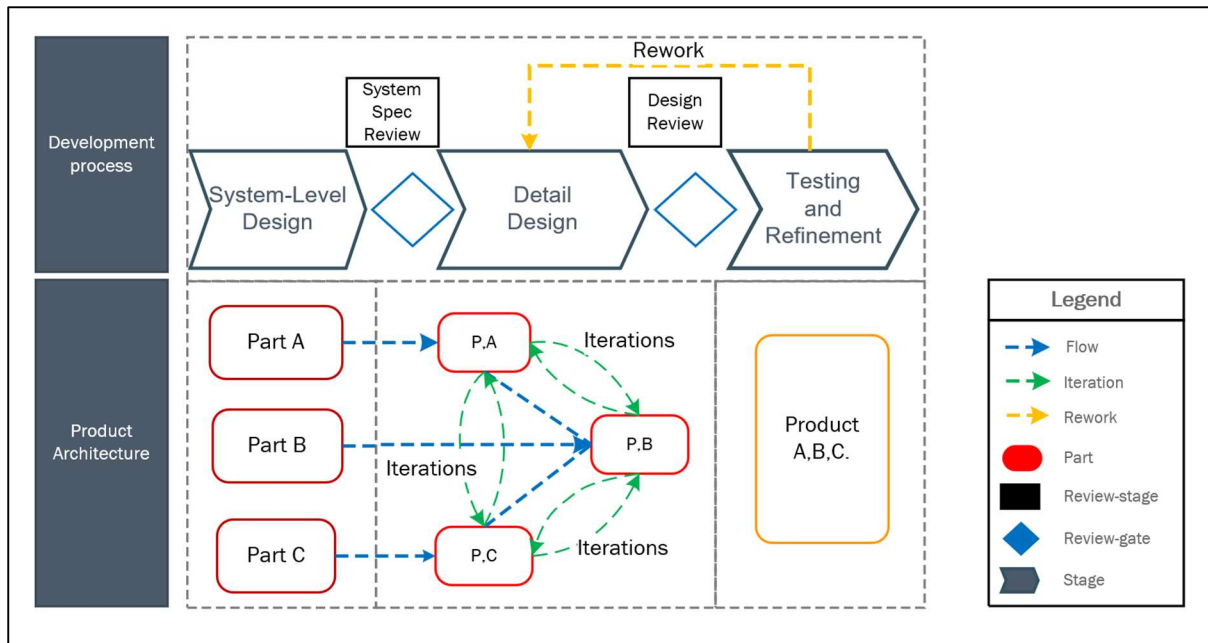


Figure 3.2.1.1 Cross-gate's product development and architecture design and three in-stage iteration feedback loops.

The chronological structure (the process) and the corresponding product architecture (the product) delineate specific in-stage feedback loop iterations of individual designers and design teams (the people) [Green arrows]. Cross-gate feedback loop iterations in rework [Yellow arrows] influence the product development performance and outcomes.

3.2.1 Identify contributing theoretical bases prior to technological advances.

The identified theoretical base considered that the social processes involving team-working, complex problem-solving, creativity and information exchange characterise product development processes as complex socio-technical systems (Robinson, 2016). Those socio-technical systems are developed to perform specific tasks, including technical aspects such as technology, infrastructure, and processes; socio aspects such as people, goals, and culture and systemic connections between them (Clegg et al., 2017). Within the design process, designers communicate their ideas by different means and documents. The uncertainty introduced during designers' understanding and ideas evolution requires more exploration of alternatives, making the process iterative (Piccolo et al., 2019).

The theoretical bases also consider the three representations of the iterative process suggested by Wynn (2007): Task-based, information-based, and actor based. The task-based approach depicts the engineering design process as tasks are attempted and revisited until design completion. In this approach, iterations are viewed as already finished tasks or the execution of similar tasks in different contexts. Task-

based approaches are information processing approaches that create and refine information about the product. In contrast, information-based methods decompose the task into multiple iterations enabling feedback by releasing preliminary information before its completion (Wynn, 2007); in an information-based approach, the information about the process determines the process behaviour. Actor-based perspectives view the design process as social coordination of independent actors negotiating trade-offs. Where design iterations result from a continuous dialogue between them, this perspective allows the representation of multiple process participants and their self-organised behaviours (Wynn, 2007).

A literature review was used to identify prior technological advances as similar digital tools. The analysis identifies relevant technological advances that have an essential impact on the design simulation of the engineering design process. However, a relevant trend in studying simulations of engineering design teams has focused on the actor-based perspective. It identifies tools for simulating and supporting teamwork behaviours to improve team effectiveness, profiling or examining special features on team performance and modelling collaborative product development (Perišić et al., 2016). However, several models ignore social interactions like communication patterns or other social characteristics (Perišić et al., 2016). This research considered communication patterns as an essential component for the simulation of the engineering design process.

3.2.2 ADR Stage: Building intervention and evaluation.

The second stage of ADR uses the problem framing and theoretical premises from Stage 1. It provides a platform for generating the initial design of the IT artefact, in this thesis, the conceptual framework, which was further shaped by organisational use and subsequent design prototyping cycles (Sein et al., 2011).

3.2.2.1 Discover the initial knowledge-creation target.

The shaping of the artifact requires interaction between technological and contextual dimensions, and the interaction is manifested in knowledge-creation targets (Sein et al., 2011). This research takes the emerging computer simulation approach that captures the complexity of socio-technical systems and provides a comprehensive and holistic view. It uses simulation models that represent real-world or anticipated systems, such as a design concept or a design process, which reflect the outcomes of the real-world systems for experimentation and understanding of the system's performance. The research aims to develop a conceptual framework for engineering processes to support design management and the overall knowledge-creation target.

This study used the Djanatliev and German (2015) method to guide the prototyping/development of domain-specific hybrid simulations. The framework provides a macro-structure for implementing simulation models, suggesting three significant processes to structure the simulation scope. First, independent levels of abstraction or views on the system are identified. Second, the explanation of how simulation models are linked to the abstraction levels and how the simulation paradigm is used to model structures at the considered level is developed. Finally, connections that reflect the interaction between abstraction levels are identified and defined. The definitions of how paradigm-related elements are connected and inform how the interaction between simulation paradigms is realised.

Identifying abstraction levels is possible by dividing the overall domain scope into specific subclasses. Independent simulations can be implemented through a hierarchical breakdown where macro, meso, and micro levels cover the actual situations within the simulation context. Explaining how simulation models are linked to the abstraction levels is essential to determine: the relevant continuous structures if they are present, if processes must be traversed, or if the representation of individual behaviours is necessary (Djanatliev & German, 2015). The stage-gates model provides the chronological structure (the process), which, for the design of a given product, is combined with a product architecture to form the development process structure. A case study was used to exemplify the engineering design process, identifying systems and subsystems of the designed product.

3.2.2.2 Execute building intervention-evaluation (BIE) cycles.

The outcome of this stage is the realised design artifact (Sein et al., 2011). Furthermore, the BIE cycle involves the formulation of general design principles.

In the development of the system analysis, the product development processes compromise the product architecture, the social and organisational patterns of the actors, the logical structure of the process development, iteration feedback loops, and rework. Djanatliev and German's (2015) frameworks suggest using abstraction levels for domain (i. e., macro, meso and micro) levels in identifying the system elements. At the macro level, fewer details and high abstraction levels are required, and a holistic view can be achieved; however, this perspective is not suitable for the intended socio-technical resource of this work. On the other hand, at the meso-level, models cover tactical-level interactions such as product development processes. The product architecture is used to identify the parts and components of the product being designed. Micro-level models cover operational-level interactions in fine detail. Here, agent-based models represent the interaction between designers identifying

actions, states, and behaviours. Figure 3.2.2.1 shows the abstraction levels and the corresponding simulation methods in the correspondent abstraction level suggested by Djanatliev and German (2015).

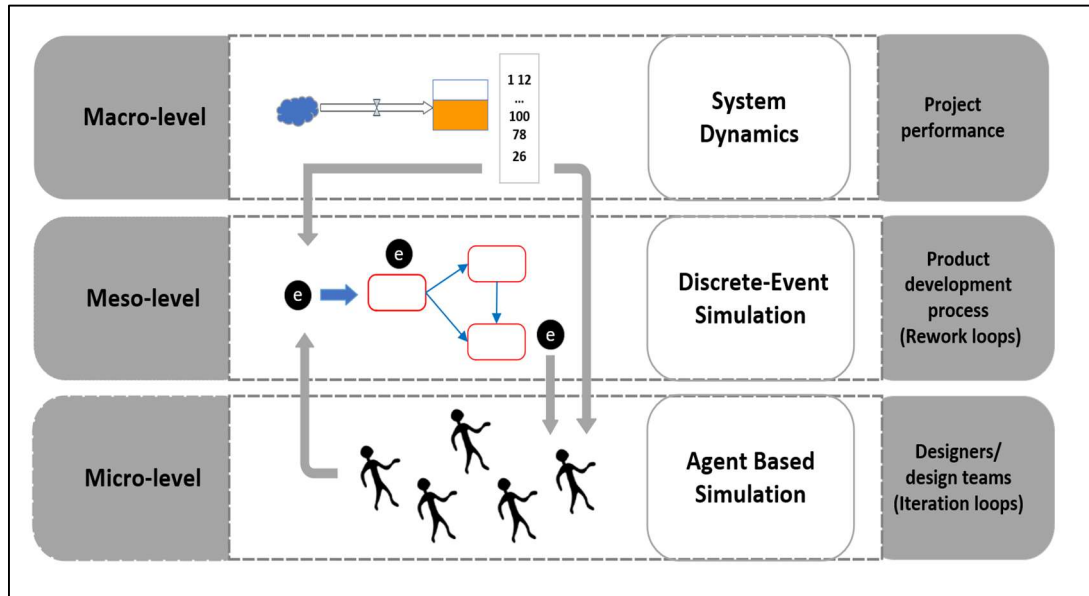


Figure 3.2.2.1 Abstraction level identification Djanatliev and German (2013)

3.2.2.3 Asses the need for additional cycle repeats.

The ADR method visualises the research process as a mix of interrelated activities of building the IT artifact, intervening in the organisation, and evaluating it concurrently (Sein et al., 2011).

The previous evaluation simulation experiment identified the requirements for the following research stage: the development of agent-based models that capture iteration as micro-level feedback loops in the agent-based model. Iterations and rework are triggered by the states and behaviours of the agents (designers) sharing and processing information. Using the framework suggested by Djanatliev and German (2015), the abstraction levels or views of the system this research established that at a meso-level, the product development process and the product architecture help identify the possible structure of the simulation model. The product development process's linear logic matches the consecutive progression of tasks that the discrete events model allows representing.

Micro-level models cover operational-level interactions in fine detail at this abstraction level; agent-based models represent the social interactions between identifying actors, behaviours, states, and relationships.

The conceptualisation of those activities entails differentiating between information processing and communication activities. Communication and coordination mechanisms of the actor influence information processes and are determinants for the successful progression and conclusion of the design work and outcomes (Montagna and Cantamessa, 2017; Hoegl and Weinkauff, 2005). Engineering designers expend 40 to 66% of their working time processing, communicating and disseminating information (Robinson, 2010). Information processing includes seeking information, gathering information, processing information, and evaluating information. At the same time, communication processes include asking questions and answering questions.

3.2.3 ADR Stage reflection and learning

The reflection and learning stage moves conceptually from building a solution for a particular instance to applying the learning to a broader class of problems (Sein et al., 2011).

3.2.3.1 Reflect on the design and redesign during the project.

The cycles of design and redesign of the simulation models allowed the identification of three critical stages of the design process to simulate the feedback loops in the engineering design: the design activity, the information processing state, and the communication state. The simulation of communication process should be simulated as a group of constructs, not as a single block in the simulation. The iteration feedback loops result from evaluating and reformulating the questions in the simulation model. The rework cycles are determined for the simulation decision on continuing with incomplete or missing information.

3.2.4 ADR Stage: Formalisation of learning

Researchers outline the accomplishments in the IT artifact and describe the organisational outcomes to formalise the learning (Sein et al., 2011).

3.2.4.1 Articulate outcomes as design principles.

Articulation of design principles suggests that: the development of simulations of complex product development socio-technical systems will involve the identification of a general-purpose product development process framework providing a chronological structure and the product architecture identifying parts being designed. The actors living in the system, their relationships, behaviours, interactions, and possible states will also be identified, as will the abstraction levels or views of the system establishing boundaries and system elements. Additionally, connections or interaction points between abstraction levels will also be identified. Such simulation

models could delineate specific iteration feedback loops of individual designers or design teams or process development feedback loops as a form of rework.

3.2.4.2. Sub-task: Generalisation of design principles.

This research used the concept of analytical generalisation suggested by Cronholm and Göbel (2022). That suggests that the analytical generalisation approach seeks to expand theories beyond their current domain (Yin, R.K., 1994), involving a reasoned judgement about the extent to which the findings from one study can be used as a guide to what might occur in another situation (Kvale, 2012).

The development of the framework for the simulation of feedback loops in engineering design uses the core structure suggested for constructing hybrid simulation models of Mykoniatis and Angelopoulou's (2020) conceptual model, development process, verification and validation, and results and documentation. However, due to the Mykoniatis and Angelopoulos framework approach focusing heavily on the simulation's instrumentation details, this research introduces the framework of Nikolic and Ghorbani (2011) to implement a conceptual model supported by a socio-technical perspective. This framework suggests five principal stages for the development of the simulation of complex socio-technical systems: (1) system analysis, (2) model design, (3) detailed design, (4) software implementation, and (5) model evaluation.

The system identification stage of Nikolic's and Ghorbani's (2011) framework was implemented developing an inventory of the system components that identify explicitly: (a) relevant concepts, (b) actors or objects, (c) relevant behaviours, (d) interactions of flows (continuous or discrete), and (e) states or properties.

The actors are entities capable of decision-making; all others are considered objects. The properties that describe and specify agents are states. Interactions are in-out interplays between agents, and the behaviours are state changes due to interactions. The concepts of actors, objects, behaviours, interactions, and states emerge from the data collection and discussions. Only explicitly stated ideas can be compared to the later implementation of the model or theoretical models in the literature (Nikolic and Lukszo, 2013).

Agents are the basic units of the model, representing one or more actors in the system. They are recognised by their boundaries, states, behaviours, and interaction ability. Interactions are the ways through which agents affect each other. Interactions may be short-term or long-lasting, immediate, or delayed. In this study, the criteria for structuration of the inventory are presented in table 3.2.4.2.1.

Table 3.2.4.2.1 Criteria for the identification of system elements, based on Nikolic and Lukszo (2013).

1. Given the inventory results, consider actors and objects with useful boundaries (physical, organisational, and functional). Entities capable of independent decision-making will be the agents, and all others are considered objects. Agents can contain or interact with objects (such as companies owning facilities or a postman processing a letter).
2. Within the entities defined above, identify properties that describe and specify the agents. These are states.
3. Within the entities defined above, search for interactions with agents or objects outside, both incoming and outgoing. These are interactions.
4. Within entities defined above, identify state changes caused by interactions or other state changes and state changes that lead to interactions or other state changes. These are behaviours.
5. Note which agents, interactions and behaviours are dynamic and which are static and at what time frame.
6. If necessary, organise agents hierarchically by ordering them in a nested way, as a box within a box (e.g., departments within a company).

After the inventory and structuration phase, the identified system components were formalised, developing ontologies where concepts, including objects and other entities that are assumed to integrate the system and the relationships between them, are formally *encoded* (Gruber, 1993). Ontologies seek to minimise misunderstandings and are meant to be computer-understandable but accessible to human users (Nikolic and Ghorbani, 2011).

This thesis used two ontology approaches for the conceptualisation of the system. First, the classes and instances identification approach were used, where a class is a generalisation of several instances, and an instance is a single identifiable object within the limits and scope of the model. Second, the ontology suite of the product life cycle developed by Otte et al. (2019) was used to formalise material entities, processes and information entities and their relationships.

The logical model formalisation of the conceptual framework for simulation of feedback loops in engineering design proposed in this thesis is deployed and complemented with activities suggested in Nikolic and Lukszo (2013) for the model formalisation stage. This thesis introduces the model narrative with an analysis of the

possible variables for the simulation and is presented in a graphical representation. The pseudo-code is also presented, including the identified system elements and possible initial parameters for the implementation.

The following stage, in the framework of Mykoniatis and Angelopoloulou (2020), requires the selection of the simulation method in the implementation process stage. This thesis used the framework developed by Djanatljev and German (2013; 2015), which suggests three steps for the definition of the simulation approaches for the implementation of hybrid simulations: (1) identifying independent levels of abstraction or views of the system, (2) linking the simulation paradigms to the identified abstraction levels, and (3) describing how paradigm elements can be connected, for which this research returned to Mykoniatis and Angelopoulos' (2020) framework identification and classification of the interaction points. The identification of interaction points results from mapping the boundaries of the models that need communication (Mykoniatis and Angelopoloulou, 2020). The interaction points are pairs of information exchange between the models that are correctly "captured by" or "influenced by" each simulation modelling approach.

The Mykoniatis' and Angelopoloulou (2020) framework suggests two main information exchange categories and their subcategories. First, the value assignment relationships include mathematical formulations and the replacement of values between equivalent variables. This category includes three subcategories: (1) direct replacement value of variables for equivalent variables of information exchange; (2) the interaction points that seize values of information exchange that need to be aggregated (accumulated) or disaggregated from one model to equivalent values of the other model; (3) causal relationships that are interaction points described explicitly with mathematical relationships. On the other hand, impact statement relationships cannot be expressed using values; these relationships are related to abstract concepts and is presented in table 3.2.4.2.2.

Table 3.2.4.2.2 Types of value assessment relationships from Mykoniatis and Angelopoulou (2020).

Category	Types of relationship	Description
A.Value Assignment	Direct replacement of values of variables	Corresponds to interaction points that represent equivalent variables of information exchange in both models.
	Aggregation/dissagregation	Corresponds to interaction points that seize values of information exchange that need to be aggregated (accumulated) or disaggregated form the one model to equivalent values of the other model.
	Causal relationships	Correspond to interaction points described by explicitly mathematical relationships.
B. Impact Statement	Add/remove inject agents entities	Inject agents, in any point of the simulation process.
	Control flow	Correspond to "if" "for" and "while" statements and define the flow of a particular logic.
	Trigger event	Could be "timeout" "message", "condition" "rate" and arrival.
	State-chart control	Correspond to the state that may control the flow among two models, update variables from other models or trigger other relationship.

Mykoniatis and Angelopoulou (2020) identified three subcategories for impact statement relationships: (1) add/remove/inject/transfer agents or entities; (2) control flow relationships, which correspond to “if,” “for,” and “while” statements and define the flow of a particular logic; (3) trigger event relationships, timeouts, messages, conditions, rates, and arrival triggers.

The framework’s design principles guide the development of the conceptual framework. However, during the BIE cycles, the development of the conceptual framework is guided by emerging design principles when identifying the structural components of the conceptual framework.

3.4.2.3 Sub-task: Formulation of design principles

There are four relevant characteristics for constructing simulation models of feedback loops in engineering design. First, the product architecture that identifies parts of the product being designed and suggests the possible structure of the design process. Second, the process workflow which considers the phase's completion as discrete events. Third, the social interaction between designers which allows identifying actions, states, and behaviours to characterise actors and agents. Fourth, the definition of interactions between simulation methods which defines the interplays between simulation models.

Using Nikolic and Lukszo's (2013) criteria for identifying the system elements, the system's conceptualisation activity identifies actors, relationships, behaviours, and possible states. In the design case study, the actors are designers performing a design.

3.3 Case study

The socio-technical systems approach to engineering design of this thesis suggests that the product architecture, which identifies individual parts that need to be designed and infers the development process structure, determines a scope for the conceptual framework for the simulation of feedback loops in engineering design, and must include the social interactions (actions, states, and behaviours) of designers used to inform the design task and activities for each part.

Bicycles are machines with hierarchies of subcomponents and complex part dependencies (Regenwetter et al., 2021) the division of a bicycle design solution into subsystems allows a clear view of the components of the whole system its parts and relationships (Boessenkool and Meijer, 2013). An analysis of the bicycle architecture development resulted in the identification of three interacting systems that could be used in the conceptual framework implementation used in the case study.

- 1) Systems behaviour. Identifies the system functions, which makes the system behave in specific way.
- 2) Flow and control. Identifies the functional relationships and describing how the functions are related.
- 3) Systems structure. The physical architecture of the system, providing a clear view of the finished system/product and its components. (Boessenkool and Meijer, 2013) as is shown in figure 3.3.1.2.

From the systems decomposition of the physical architecture of bicycle the handlebar assembly structure, shown in figure 3.3.1.3, is selected to configure the case study because identifies four required teams (or agents) to design individual parts and its integration on the handlebar assembly subsystem, configuring the product architecture required.

The hypothetical description of the design process for those four components, during the systems level and detailed design stages, in a stage gates model processes infers the development process, as is shown in figure 3.2.1.1, in section 3.2.1 of this chapter, configuring the product development process required to structure simulation models. The social interactions and information behaviours of designers

during the design processes are described in the narrative of the design process, as communication and coordination activities and decision-making process over the information.

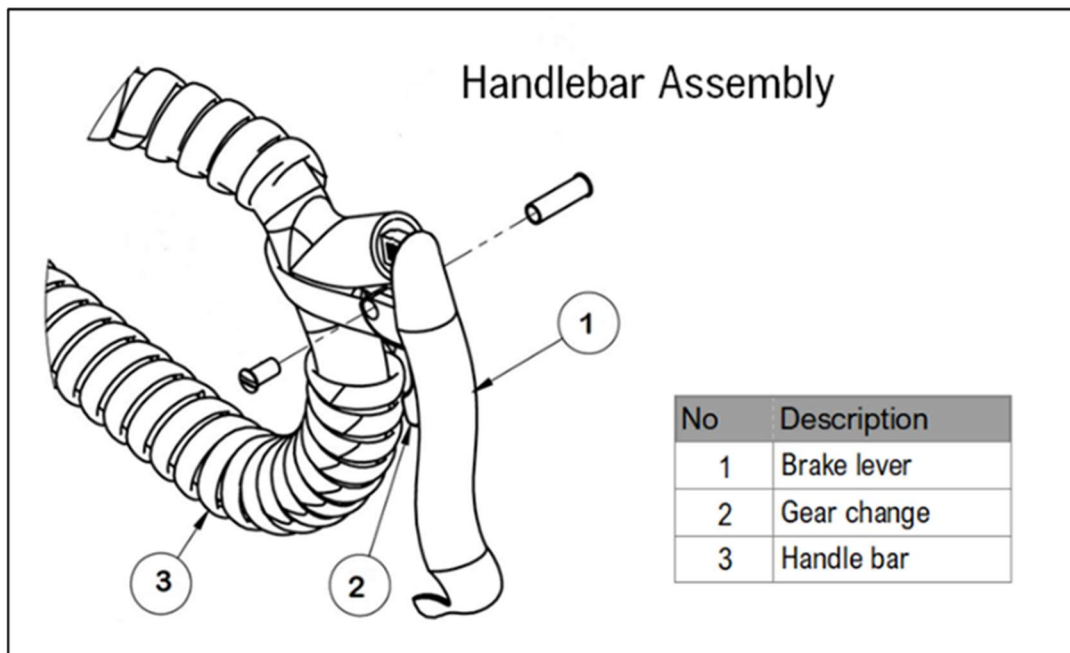


Figure 3.3.1.1, Handlebar assembly and parts identification.

3.3.1 Bicycle design.

A significant trend in the literature is in the design of frames and frame elements, supported by various frame designers and manufacturers, with an established design philosophy that suggests that the intended purpose and rider's fit are the prime objectives of all bicycle designs. The rider's fit term is generally associated with the application of anthropometric parameters. Figure 3.3.1.1 depicts the major frame design parameters of a typical modern-day bicycle.

The principles of bicycle design have been explored since the earliest predecessors to the modern bicycle. Nowadays, bicycle design optimisations are a well-researched field where significant effort is dedicated to improving the aerodynamics and structure of the bicycle, and exploring the sizing and fitting practices using the wide availability of anthropometric data (Regenwetter et al., 2021).

In the design of the bicycle, it is noticeable that power systems, including crank arms, pedals, chain-rings, gear-change (derailleur), brake systems, steering and handlebar system and wheels are considered necessary but not determinant element of the design intent.

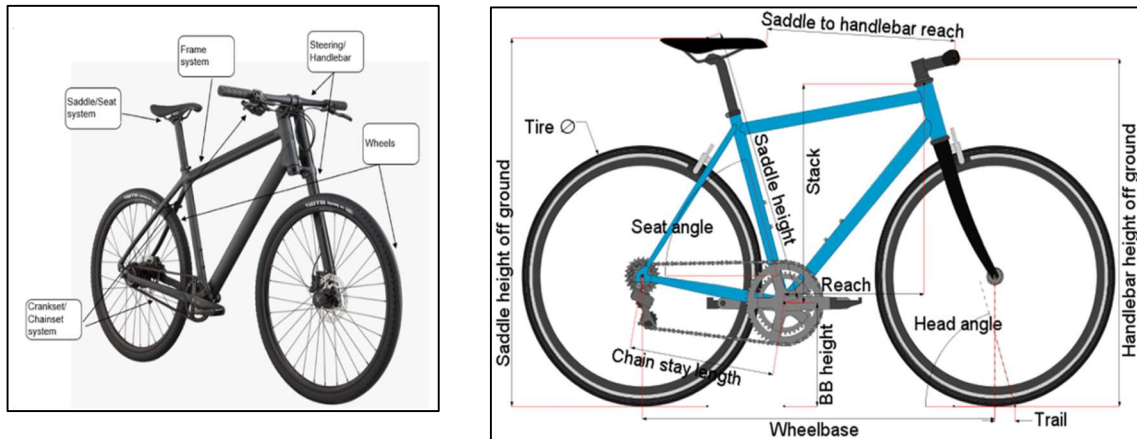


Figure 3.3.1.2 On the left side, the Cannondale bicycle and systems identification diagram from Cannondale (2020); on the right side, the leading bicycle dimensions for frame construction from Sulmall (2022).

The diagram in Figure 3.3.1.3 shows the hierarchical integration of the bicycle parts; from there, the system decomposition of the handlebar assembly identifies the four teams needed to design the subsystems: (1) the brake lever, (2) the gear change, (3) the handlebar, and (4) the handlebar assembly as the target system of this study.

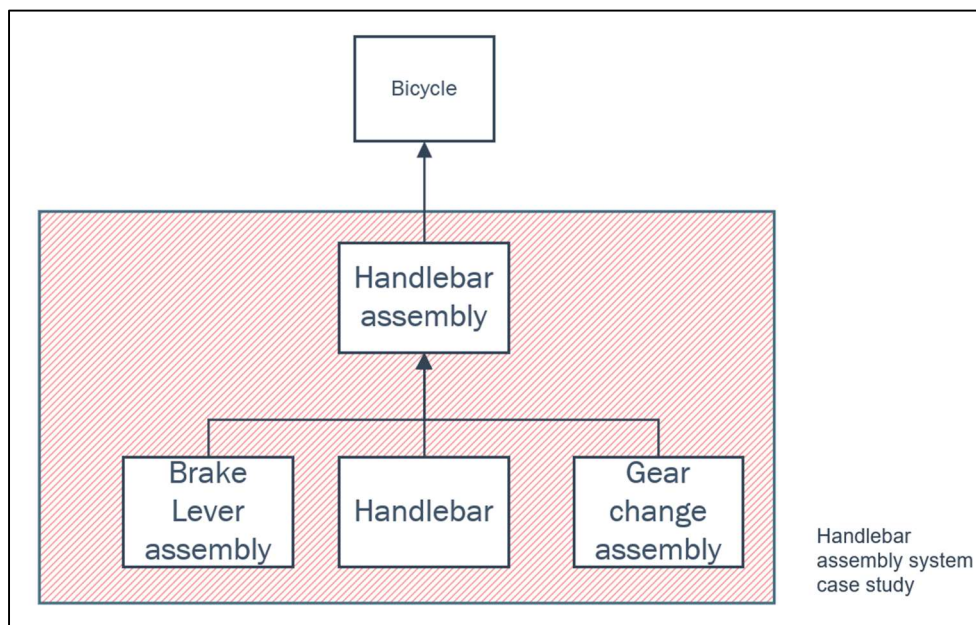


Figure 3.3.1.3 Structure diagram from the bicycle design architecture.

The description of the hypothetical process of the designing of the handlebar assembly, is as follows: The (1) bicycle design development process starts when a

(2) design request is delivered simultaneously to the (3) brake lever, (4) gear change, and (5) handlebar designers, and the (6) handlebar assembly integration designer, who must wait for the three individual (7) part designs to perform their process. Each (8) designer iterates the design for each (9) component and (10) communicates with each other (11), asking and (12) answering questions. In some cases, (13) feedback loops are (14) coordinated (i.e., (15) communicated effectively and (16) on time, while in other cases, they are not (i.e., (17) not communicated, or (18) communicated with a delay, or (19) incomplete information). So, the designers must (20) decide to ask again or (21) carry on without the proper knowledge. Here, (22) not coordinated iterations lead to (23) rework, and (24) not coordinated rework might lead to further rework, in a (25) vicious circle.

The scope of the conceptual framework for simulating feedback loops in engineering design considers both technical and social aspects of the product development process. Aims for the identification of the key elements that enable the construction of simulations that capture engineering design processes, technical processes, and social aspects of the product development processes. To reflect the organisational and development process characteristics that mirror the product architecture, influencing the structure of the development activity.

The handlebar system design process, used in the case study, allowed the identification of the relevant components of the socio-technical system: product, processes, and social aspects of the development system. The division of a bicycle design solution into subsystems allows a clear view of the components of the whole system its parts and relationships (Boessenkool and Meijer, 2013). The hierarchical bicycle structure allows the identification subcomponents and its complex part dependencies (Regenwetter et al., 2021).

(1) The product architecture system elements identified are the handlebar part, the brake lever part, the gear change part, and the handlebar assembly which at the same time aid in the identification of necessary design teams for the development process.

(2) The social aspects are the individual designers for each part and the assembly designer. Those designers perform: The design tasks, consisting in the performance of design processes and iterations. The design activities which are the communication behaviours and information processes: Designers asking and answering questions and sending, receiving, and evaluating information.

(3) The development processes identified are the systems level design and the detailed design stages from the stage gate process model used in this study as is shown in figure 3.2.1.1, in section 3.2.1 of this chapter.

The expected results of the analysis, consisting in the time to conclude the individual and the assembly parts, including the number of iterations and rework resulting, from the different configuration of parameters, are developed and presented in chapter 5, section 5.3 Validation.

3.4 Summary

This chapter provided the overall framework and research design adopted. The ADR method supports research projects by developing IT artifacts shaped by organisational contexts. Its design domain involves technical and social elements and their relationships (Dresch et al., 2015).

To summarise the chapter's content, this study answers the questions formulated in the problem formulation section, sub-section 2:

The first question is: *How do iteration feedback loops influence the product development processes?* This question was answered by establishing that, within the design process, designers communicate their thoughts by different means and documents. The exploration of alternatives during the design process, where designers develop understanding and their ideas about the solution, evolves, introduces uncertainty, and makes the process iterative (Piccolo et al., 2019). Iteration feedback loops positively impact design outcomes, improving the final design's quality or allowing the final solution's refinement (Tapia et al., 2021). Usually, adequate, accurate, and delivered on time information will result in a positive feedback loops iteration a virtuous circle forming in-stage cycles that lead to find the best design solutions (Meboldt et al., 2013).

In contrast, feedback loops resulting from rework are in-stage iterations leading to feedback-learning cycles positively affecting the final design (Tapia et al., 2021). However, those negative feedback loops, or vicious circles result from new information, or inadequate, inaccurate, or miss-coordinated information, makes necessary modifications to previous activities, completed in an earlier phase.

The second question is: *To what extent have the process, the product, and the people interactions influenced product development?* This question was answered by establishing that: engineering designers expend 40 to 66% of their working time processing, communicating, and disseminating information (Robinson, 2010).

Information processing includes seeking information, gathering information, processing information, and evaluating information. At the same time, communication processes include asking questions and answering questions. The determinant factors for the successful progression and conclusion of the design work and outcomes, are the communication and coordination mechanisms of the actors involved in the product development process (Montagna and Cantamessa, 2017; Hoegl and Weinkauff, 2005) performing the activities related to engineering design processes.

The last question is: *What are the main system characteristics representing those system elements?*

A feedback loop is a situation where two or more dynamic systems are interrelated, influencing each other in a strongly coupled circular argument (Åström and Murray, 2021). In product development, feedback loops are small or large recursive cycles that characterise relationships and iterations (Kline and Rosenberg, 1986) between product development processes and participants, emerging during engineering design information processes due to product development coordination and collaboration activities of designers and design teams (Wynn and Maier, 2022). Positive feedback loops that contribute to the quality of design within the design stage or negative feedback loops when the new information requires modifications on previous activities considered already finished from an earlier phase.

The product development process's main characteristic is its chronological structure, combined with the corresponding product architecture, which is the identification of the parts being designed, inferring the structure of the design process and the workflow, and the social interaction between designers identifying actions, states, and behaviours. Finally, the articulation of design principles suggests the following. For the development of simulations of complex socio-technical systems, the identification of a general-purpose (1) product development process framework provides a chronological structure based on the (2) product architecture, which includes identification of the parts being designed. The design principles' articulation also includes identifying (3) actors operating in the system, its relationships, behaviours, interactions, and possible states. Moreover, the abstraction levels or views of the system are used to establish boundaries and system elements. Furthermore, identifying connections or interaction points between the design process and design activities informs the interplays between simulation models. Such simulation models could delineate specific iteration feedback loops of individual designers, or design teams, or process development feedback loops as a form of rework.

The conceptualisation of the articulation of the design principles is represented in the following grid diagram, where the vertical axis represents the abstraction levels or views of the system. The horizontal axis shows the simulation methods: Agent-based (ABS), Discrete events (DES), and System Dynamics (SD). This thesis has established that the System dynamics approach is out of the scope of the study.

At the meso-level, the product architecture identifies individual parts to be designed, infers the product development process, and derives a design process workflow, determining the discrete event model. In parallel, the social interactions, actions, states, and behaviours between agents (designers) specify the agent-based simulation model of the design activities for each part of the product architecture.

Chapter 4: Conceptual framework development

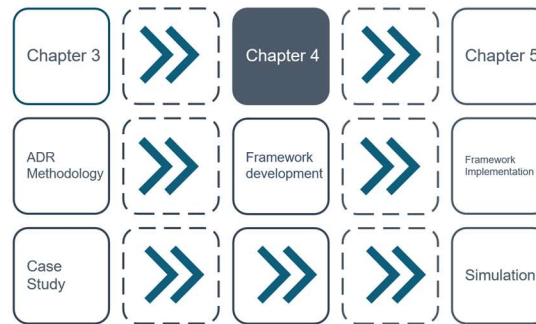


Figure 4.1.1 Chapter layout

Conceptual modelling activity focuses on capturing a representation of human perceptions of the natural world to produce a relational or logical representation of a system to be used in an information system, representing the system in the form of diagrams or models (Pastor, 2016).

The construction of the conceptual framework for the simulation of feedback loops in engineering design is supported by the framework proposed by Nikolic and Ghorbani (2011) and complemented by the framework by Nikolic and Lukszo (2013) for the simulation of socio-technical systems. The implemented conceptual framework for the simulation of the feedback loops in engineering design in this thesis is integrated with four stages: The problem formulation stage. The system identification stage. The system conceptualisation stage and the model formalisation stage, where the conceptual model is deployed.

In the Nikolic and Ghorbani (2011) framework, the problem formulation requires establishing the problem to be addressed, understanding the "problem" as a poorly understood situation that appears to have a suboptimal performance or the realisation of a lack of knowledge about a system, its behaviour, or its response to interventions. Followed by system identification, which consists of identifying the system's boundaries, and internal structure recognised in the problem identification. System identification consists of an inventory of the system components, explicitly identifying relevant concepts, actors and objects, appropriate behaviours, interactions or flows, and states or properties (Nikolic and Ghorbani, 2011). In this stage, the specified system elements must be structured, establishing which concepts are agents, states, interactions, or behaviours; the structuration activity in this thesis was performed using the criteria suggested by Nikolic and Lukszo (in Section 3.2.4 "Criteria for identification of system elements" in Chapter 3).

The third stage corresponds to the system conceptualisation, which aims to formalise the identified concepts. In this stage, the model of the world is made explicit, formal, and simultaneously computer and human understandable. This stage uses ontologies to formally “encode” the existing entities of the system to reduce ambiguities and confusion about the system’s logic and concepts used.

The model formalisation stage includes analysis for identifying variables for the simulation and developing the graphical representation of the conceptual framework for the simulation of feedback loops in engineering design. This stage consists of a pseudo-code prescribing the simulation constructs and possible parameters for the implementation.

This chapter follows the logical structure of the conceptual modelling activity and is developed as follows: Section 4.1, Problem identification, followed by 4.2, System Identification. Section 4.3 presents the System conceptualisation, followed by section 4.4, Model formalisation deploying the conceptual framework for the simulation of feedback loops in engineering design; the last section includes the summary of the chapter.

4.1. Problem identification

This stage is expected to increase the understanding of how the system works and give insight into how certain aspects influence the system’s behaviour (Nikolic and Ghorbani, 2011).

This thesis has established that product development systems are organisational systems characterised by transforming inputs from design requirements into outputs in the form of products or services released to their environment (Pessôa and Trabasso, 2017). Product development systems are integrated by intra and extra-organisational contexts (de Weerd-Nederhof, 1997). The extra-organisational context is integrated for all these entities that have no direct influence on the process but may impose restrictions or challenges; they could be governmental regulations, competitors, or suppliers. In the intra-organisational context, are allocated other organisational functions, for instance, purchasing, manufacturing or distribution de Weerd-Nederhof (1997). As well has been established that the product development system’s primary function is to execute the product development processes (Pessôa and Trabasso, 2017).

This study has also established in the conceptualisation of the research opportunity (Section 3.2.1 of Chapter 3) that product development processes are considered from an organisational perspective, a social network with multiple dimensional and

interconnected processes. Where engineering designers interact to find a design solution (Whitworth, 2009; De Bruijn and Herder, 2009; Kratzer et al., 2010; Leenders et al., 2003), these processes systematise the way the products are delivered to the environment, considering customer demands and strategic priorities. Product development processes involve a series of stage gates, where a decision to proceed or not drives the project's progression (Tapia et al., 2021).

4.1.1 What is the problem?

Socio-technical systems are developed to perform specific tasks. They include technical aspects, such as technology, infrastructure, and processes; socio aspects, such as people, goals and culture; and the systematic connections between these (Clegg et al., 2017). The new product development processes within the design domains involving team-working, complex problem-solving, creativity and information exchange are representative examples of complex socio-technical systems (Robinson, 2016). Within engineering design processes, feedback loops are perceived to increase complexity, time to market and costs. However, positive feedback loops iterations positively impact the design outcomes, for example, the quality of the final design. On the contrary, negative feedback loops resulting from rework may positively impact the final design, but their influence on project performance is high (Tapia et al., 2021).

Engineering design processes are defined as a network of activities to produce a design (O'Donovan et al., 2005) integrated into design tasks and activities. A design task is recognised as a goal-directed action, and a design activity is not necessarily so (O'Donovan et al., 2005). This thesis's design task encompasses analysis, synthesis, and evaluation activities. Design activities include the communication processes of designers or design teams, asking questions, answering, gathering, processing, or evaluating information. The product development process starts articulating a market opportunity and initialising the product development processes (Ulrich and Eppinger, 2012), which is integrated into six stages: planning, concept development, system-level design, detailed design, testing and refinement and product ramp-up. There is an evaluation gate between each step where the deliverables must be passed to proceed to the next stage (Tapia et al., 2021). The actions and activities within these processes are, in most cases, intellectual and organisational rather than physical and include developing information and formulating specifications, concepts, and design details. The product development process concludes when all the required information to support production and sales has been created and communicated (Ulrich and Eppinger, 2012).

Engineering design processes are used to develop and embed product innovations within the product development processes. Engineering design processes provide the structure for development processes projects, including the design development for the whole product development and its parts. Strategy and early design decisions influence the organisational structures needed to develop engineering designs and the social networks formed by design teams. During the design process, new information and constraints emerge, and changes in design requirements lead designers to revisit and re-evaluate design decisions, making the process iterative.

While in the multidimensional networks and highly interconnected processes that characterise product development systems, feedback loops are small or large recursive cycles defined by relationships and iterations (Kline and Rosenberg, 1986).

In engineering design, feedback loops iterations are considered to improve quality by systematically exploring and understanding the complexity of design problems (Le, H.N. et al., 2010). Feedback loop iterations result from social processes involving many parties and interactions (Bucciarelli, 1994). The communication activities, identified as the engineering designers' social and behavioural patterns, are influenced by the engineering designers' different competencies, cultures, expertise, and experience (Bucciarelli, 1988). The major challenge for the practical analysis of product development processes lies in (1) enriching the comprehension of process behaviour and (2) understanding the behaviour and characteristics of those engineering designers' communication patterns. The need for insight addressed for this study is understanding how positive and negative feedback loops resulting from human and organisational factors influence the design task performance within the engineering design processes. Within the product development processes design iterations, rework, and social interactions within and across design teams influence the progression and quality of the design task in new product development (Tapia et al., 2021). In a product development process, the stage-gate model provides a chronological structure (the process) combined with the product architecture (the product). Coordination and communications activities of the participants (designers or design teams) are the cause of feedback loops iterations. Timely and effective feedback loops lead to positive iterations. While late and not communicated, feedback loops lead to negative iterations and rework, impacting the system's performance.

4.1.3 Whose problem are we addressing?

Design managers can use the exploration of interplays between the distinct kinds of feedback loops. This can be used to inform decisions about resource allocation and

iteration utilisation to complete design tasks on time, allowing the balance between positive feedback loops in the form of design iteration and negative feedback loops in the form of avoidable rework.

4.1.4 Other actors

Other actors may include the engineering designers and project managers involved in the planning and executing the product development process.

4.2 System identification

The decomposition of the socio-technical system seeks to identify the physical and social entities of the system and the links between them (Nikolic and Lukszo, 2013). The activities of this stage involve identifying the system's internal structure. Actors, and their interactions over time, emergent patterns, system composition, and boundaries. However, due to the complexity and extension of the complex product development systems, only an interpretative and limited viewpoint can be achieved; all information gathered will contain simplifications and assumptions (Nikolic and Lukszo, 2013).

The system inventory of the product development process identified in this thesis contains: (a) concepts, (b) actors or objects, (c) relevant behaviours, (d) interactions or flows (continuous or discrete) and (e) states or properties are depicted in table 4.2.1.1. The use of the criteria for identifying the system elements in this thesis included in Section 3.2.4. enabling the structuration of the inventory and allows to establish that actors are the designers or design teams for components. Actors, exhibit behaviours of asking questions, answering questions, asking again, or even carrying on without proper information and performing a non-coordinated rework. The design activities, design brief, design information, design iterations, and feedback loops trigger interactions between actors.

Table 4.2.1.1 Inventory of concepts, actors, behaviours, interactions and properties identified.

No.	System elements	a) concepts	b) actors or objects	c) relevant behaviours	d) interactions or flows (continuous or discrete)	e) states or proprieties
1	Market opportunity	✓				
2	Physical product		✓			
3	Services		✓			
4	Environment	✓				
5	Intra-organizational context	✓				
6	Extra-organizational context	✓				
7	Suppliers		✓			
8	Competitors		✓			
9	Governmental offices		✓			
10	Regulations	✓				
11	Marketing		✓			
12	Engineering		✓			
13	Manufacturing		✓			
14	Product development process	✓				
15	Engineering design process	✓				
16	Design Tasks					✓
17	Design activities				✓	
18	Analysis					✓
19	Synthesis					✓
20	Evaluation					✓
21	Communication				✓	
22	Design teams		✓			
23	Designers		✓			
24	Asking Questions			✓		
25	Answering Questions			✓		
26	Gathering Information			✓		
27	Processing Information			✓		
28	Evaluating information			✓		
29	Planning	✓				
30	Concept Development	✓				
31	System-level design	✓				
32	Detailed design	✓				
33	Testing refinement	✓				
34	Product ramp-up	✓				
35	Evaluation-gate	✓				
36	Developing Information	✓				
37	Formulating Specifications	✓				
38	Required Information	✓				
39	Support production and sales	✓				
40	Information created and communicated					✓
41	Concepts	✓				
42	Design details	✓				
43	Production		✓			
44	Sales		✓			
45	System Architecture	✓				
46	Subsystems	✓				
47	Components		✓			
48	Development teams		✓			
49	Design Brief				✓	
50	design information				✓	
51	Designer iterates				✓	
52	Feedback loops				✓	
53	Coordinated			✓		
54	Communicated Effectively			✓		
55	On Time			✓		
56	Not communicated			✓		
57	Communicated with delay			✓		
58	Incomplete Information			✓		
59	Ask again			✓		
60	Carry on with incomplete information			✓		
61	Not Coordinated			✓		
62	Rework	✓		✓		
63	Not Coordinated rework			✓		
64	Vicious circle	✓				

4.3 System conceptualisation

The previous stage has provided a general identification of the system elements of the design process and a preliminary categorisation of the interactions, agents, states, and behaviours regarded as an inventory of the system components.

However, after completing this stage, the identified concepts are expressed in a natural language unsuitable for computers. The model of the world requires to be explicit, formal, and able to be understood by computers and human beings. The formalisation activity seeks to make the system description generalised beyond one domain.

Ontologies can be used to help in the process of formalisation of concepts and relationships without focusing on software implementations. This thesis used two approaches to the implementation of ontologies. In the first approach, an instance is a single identifiable object within the limits of the scope of the model, and a class can be considered a generalisation of several instances. This approach is used to form the basis of class structures, where the ‘is a’ relation is coded as the subclass relationship in the class description, and the ‘has a’ provides information on the properties of the class and possible values (Nikolic and Lukszo, 2013). The formalised concepts in the model shown in figure 4.3.1.1 show the class structure for the product development processes in this thesis, where the red frames are the class elements, and the black frame denotes instances of the class.

On the other hand, there is the Otte et al. (2019) suite of modular ontologies of the product life cycle and its successive phases. The modular ontology identified three main stages: (1) Beginning of Life (BOL), (2) Middle of Life (MOL), and (3) End of Life (EOL); (Shin et al., 2011). The BOL phase is integrated by planning, design, and manufacturing. In the design stage, a design action implies identifying requirements, defining concepts, and refining detailed designs, prototypes, and tests. The modular ontology of the life cycle identified material entities, information entities and processes; This ontology approach also captures the relationships with a two-place instance-level relationship. The modular ontologies generate life cycle data by involving artefacts and human agents. This thesis uses the terms defined by Otte et al. (2019) to identify the relationships between the system elements prescribing:

- Material entities are agents, elements of the natural world, which have a physical existence.
- Processes are related to planned or unplanned activities; the actions needed to achieve a goal.

- Information entities are regarded as different forms of communication, related or not to the processes or material entities (see figure 4.3.1.2).

The system elements identified in problem identification are categorised into the material, process, and information entities presented in table 4.3.1.1.

Table 4.3.1.1: Ontology concepts categorization according to Otte et al, (2019).

No.	System elements	Material	Process	Information
1	Product development		✓	
2	Design brief			✓
3	Designer	✓		
4	Design Task			✓
5	Analysis		✓	
6	Synthesis		✓	
7	Evaluation		✓	
8	Product architecture			✓
9	Part	✓		
10	Design activity		✓	
11	Communication process		✓	
12	Ask (Ask for information)		✓	
13	Answering (providing info)		✓	
14	Ask again		✓	
15	Communicated effectively		✓	
16	Design Iteration		✓	
17	Feedback loops			✓
18	Rework		✓	
19	Vicious circle		✓	
20	Not coordinated		✓	
21	Incomplete information		✓	
22	information		✓	
23	Not communicated		✓	
24	Gathering information		✓	
25	Processing information		✓	
26	Evaluating information		✓	

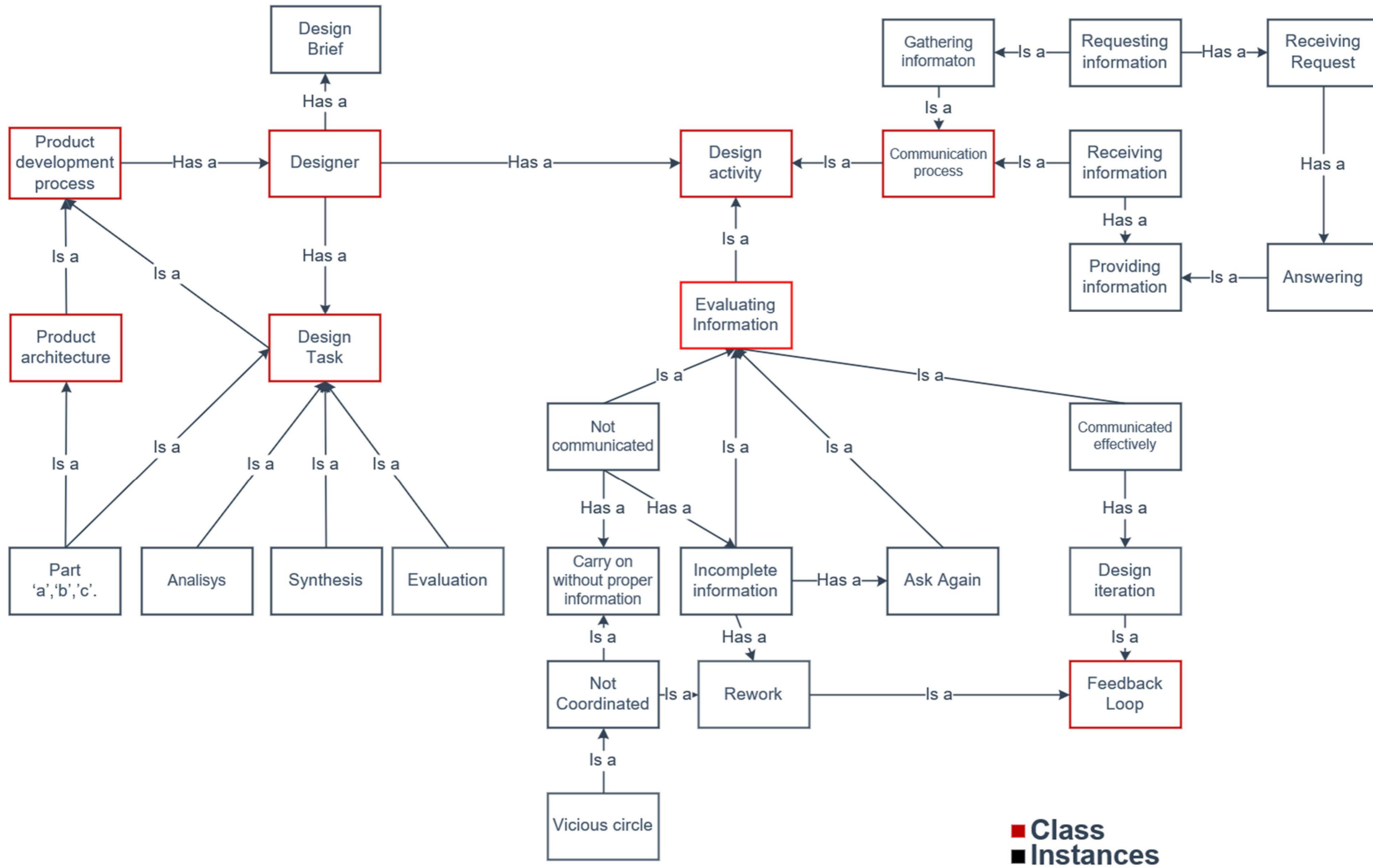


Figure 4.3.1.1 Ontology classes and instances identification.

Moreover, the relationships between material entities, processes, and information entities related to Otte et al. (2019) suite of modular ontologies of the product life cycle are a set of instance-level relations that include:

- Participates in = a primitive relationship between a process and a continuant. *
- Agent in= a sub-relation of 'participates in' that obtains when the continuant participates, and the continuant is casually active in the relevant process.
- Is input of= a sub-relation of "participates in" that is obtained when the continuant participates in the process and when the presence of the continuant at the beginning of the process is necessary for the start.
- Is output of= a sub-relation of participates in, obtained when the continuant participates in the process, and its presence at the end of the process is necessary for its completion.

The sub-relations of 'is about, including 'describes', 'represents', 'prescribes', and 'designates', relate different information content entities, such as reports, photographs, design specifications, names and the things they are about (Otte et al., 2019).

*A continuant is a material entity that endures through time. Moreover, there are two kinds of continuants; independent continuants, which are objects, their material parts and boundaries, and dependent continuants; which are attributes of material entities, including qualities, roles and functions (Arp et al., 2015).

Table 4.3.1.1 shows the system elements identified and their classification based on the ontology of the product life cycle, identifying material elements, processes, and information concepts. The ontology model developed, shown in Figure 4.3.1.1, allowed the formalisation of the concepts to be used during the simulation model construction in the implementation stage. The ontology developed using Ottes' (2019) approach, shown in Figure 4.3.1.2, enables the formalisation of the interactions between the system elements regarding processes, information and material entities.

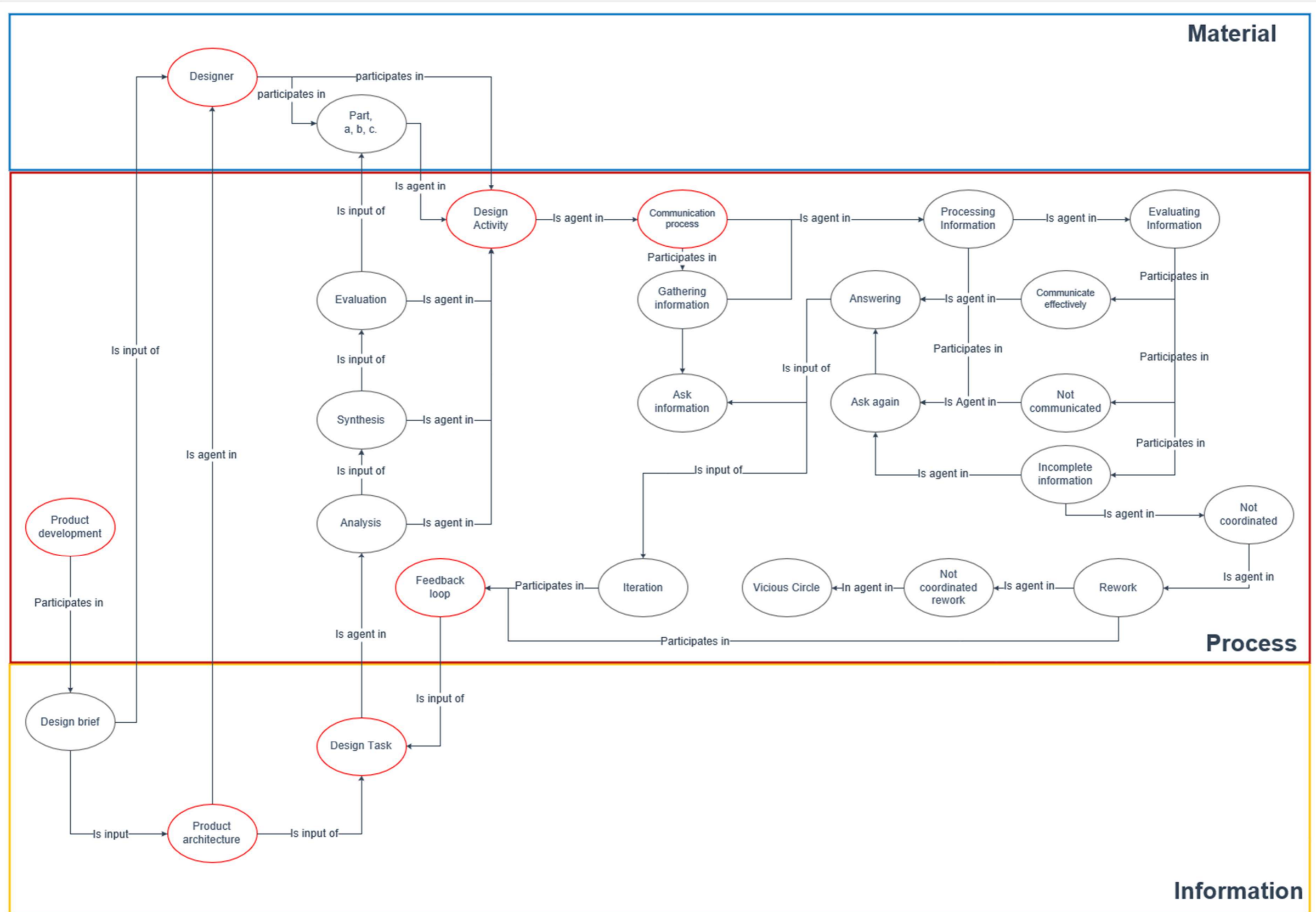


Figure 4.3.1.2 System conceptualisation using an ontological approach, based on Otte et al. (2019).

4.4 Model formalisation

During the previous stage, the identification of who and what is in the model has been achieved. In the formalisation stage, who does what and when is established.

This stage aims to map the previously identified concepts into an artificial model with an abstract representation, creating a model narrative as an account of a series of events and facts, given in order and with an established connection between them.

This includes the analysis of two relevant frameworks for the simulations of engineering teams to identify those variables useful in developing the narrative in the simulations of engineering design teams. And the logical model implementation consisted of a graphical representation of the conceptual framework for simulating feedback loops in engineering design.

Crowder et al. (2012) identified and measured nine variables and three sub-tasks, categorised into three sub-categories. Underpinning teamwork in an engineering design context namely "mental models", "communication", "trust", "learning time", "availability", "response rate", "motivation", "competency", and "workflow". Crowder et al. (2012) explain that; teams share a mental model to the extent that the team members share a common cognitive representation of the working environment and that the transfer of information between team members using various media is communication. Trust is defined as the "trust" between agents. Learning is the variable name for the time teams require to assimilate an undertaking task.

Furthermore, Time Availability measures the percentage of time individual members spend working toward team objectives. The possibility of a team member answering a request for information is the response rate. The variable motivation refers to the commitment of the individuals to the achievement of the team's goals, and competency is the ability of the team to perform their work tasks. The workflow variable is defined as the scheduling of the subtasks that make up the overall task of the team, and the considered sub-tasks can be executed in parallel, in sequence or overlapped.

Crowder et al. (2012) identified characteristics of the sub-tasks as well,

- The sub-task difficulty is an individual's required work to solve a problem.
- Sub-task quality is the overall quality of all outputs of the team.
- Sub-task working time is where higher scores indicate more time (poor performance).

In a different order, Perišić et al. (2016) suggested six agent characteristics and four project characteristics in their framework, starting with "experience", "availability", "behaviours", "motivation", and competencies. The four project characteristics are competency level/Innovation level, activity type, and resources. Their framework also identified the agent-based activities as "roles" and defined design agents and team leader agents. Their model was able to determine if agents were able to perform technical and non-technical skill activities, characterising competencies and distinguishing between specific (knowledge and skills) and social aspects. Perisic's framework also identified experience as a concept related to project efficiency, and motivation, as a complementary aspect of experience related to the agent's commitment to the project goals and behaviour as the strategy agents take when faced with problem-solving. The Crowder et al. (2012) and Perišić et al. (2016) frameworks recognise motivation, competency, and availability, whereas Crowder et al. (2012) identify learning time, Perišić et al. (2016) acknowledge experience; similarly, they recognise response rate, and behaviours. The table 4.4.1.1 shows the variables, characteristics of the agents identified their respective frameworks. The three variables not included in Perisic's framework are shared mental models, communication, and trust.

Table 4.4.1.1 Comparison of identified variables and agent characteristics from Crowder et al. (2012) and Perisic et al (2016).

Team working variables selected for Crowder,et.al. 2012.					Agent characteristics from Perisic, et.al., 2016.		
Variable	Variable type	Team-level variable	Individual-level variable	Sub-task variable	Characteristics	Project customization	
Shared mental models	Process	X			Role		
Communication	Process	X					
Trust	Process	X	X				
Learning time	Process		X		Experience		
Availability	Process		X		Avaliability		
Response rate	Process		X		Behaviors		
Motivation	Process		X		Motivation		
Competency	Process		X		Competencies		
Workflow	Process			X			
Sub-task difficulty	Process		X	X		Complexity level	Innovation level
Sub-task	Performance		X	X		Activity type	
Sub-task working time	Performance		X	X		Resources	

Modelling processes and environments that reflect different product development projects and organisations should be possible considering project factors such as activity dependency and iteration. These may drive workflow creation and influence its structure. The teamwork process's complexity suggests the use of a wide range of variables for constructing conceptual and simulation models. However, a limited

number for inclusion based on theoretical and empirical criteria has yet to be selected in this thesis.

As a result of the previous analysis in in the systems identification and systems conceptualization, this research identified six variables and characteristics useful in the development of the conceptual framework resulting from the conceptualisation and ontology constructions are: (1) Product Architecture, (2) Designer, (3) Design Task, (4) Design Activity including communication processes, (5) Feedback loops, and(6) From the product development processes only system level and detailed design are considered in this thesis the summarized identification of variables and characteristics is presented in table 4.4.1.2.

Table 4.4.1.2. Variables and characteristics identification. (The product development process included is based on Ulrich and Eppinger, 2012.

Variable/ Characteristics	Activity	Task	Info direction	Recipient status		
Product architecture	Functional and physical elements are identified	The product architecture design allows the assignation of detailed design and testing activities to teams or individuals.	In/Out	Active		
Designer	Design team or designer	Performs design task	In/Out	Active		
		Performs design activity	In/Out	Active		
Design Task	Analysis	Problem and solution interrelations, 38% of average time	In/Out	Active		
	Synthesis	Generation of design Solution, 54% of average time	In/Out	Active		
	Evaluation	Assessing the validity of solutions, 8% of average time	In/Out	Active		
Design activity	Communication processes	Gathering information (Requesting Info)	Ask for information	Out	Active	
			Ask again	In/Out	Active	
		Evaluating information	Disseminating information	Provide information (Answer to questions)	In/Out	Active
			Not communicated	In/Out	Passive	
			Incomplete information	In/Out	Passive	
			Carry-on	Out	Active	
Communicated effectively	Out	Active				
Feedback loop	Design iteration	Well communicated information	In-Stage Iteration between within stages with zero impact in previous decision	In/Out	Passive	
	Rework	Not communicated/ Incomplete information	Cross-gate iterations, iterations happening after a decision, affecting previous decisions	In/Out	Passive	
Product development process	Planning	Opportunity identification	Target market, business goals, key assumptions and constrains	Out	Active	
	Concept development	Generation of product concepts	Generation and evaluation of concepts based on identified market needs	In/Out	Active	
	Decision gate	Evaluation to proceed to next		In/Out	Passive	
	System-level design	Decomposition of the product into subsystems and components	Preliminary design key components and allocation of detail design responsibilities	In/Out	Active	
	Decision gate	Evaluation to proceed to next		In/Out	Passive	
	Detail Design	Specification of the geometry, materials and tolerances	Complete specification of all unique parts in the product	In/Out	Active	
	Decision gate	Evaluation to proceed to next		In/Out	Passive	
	Testing and refinement	The construction and evaluation of multiple pre-production versions	Alpha and Beta versions are constructed with the intended production processes	In/Out	Active	
	Decision gate	Evaluation to proceed to next		In/Out	Passive	
Product Ramp-up	Product is made using intended product system	Products are carefully evaluated to identify remaining flaws	In/Out	Active		

Team working is related to several desirable organisational outcomes, such as efficiency and improved quality; teams enable organisations to develop high-quality

ideas and products efficiently and effectively. Team working environments require members to communicate and collaborate across disciplinary, departmental, and company boundaries (Crowder et al., 2012). The team process theory describes the interdependent activities of team members as a single interwoven process (Cash et al., 2019) that can be described in terms of the activities involved and how they vary concerning the goals, underpinning actions and context. The design activity operationalises these elements concerning the design work (Cash et al., 2015). Teamwork does not result simply from aggregating the behaviour of individuals, nor can its outcome be measured at the scale of individual units (Perišić et al., 2016).

Martinec et al (2017) points that team activities underpinning team processes can be addressed by one or more team members, allocated or distributed with one specific project, in a cross-functional project team or with a particular design phase as in a functional team. As team members move between the team and taskwork and address various sub-goals, the individual activity threads intertwine to form the team processes. The teamwork activity is a set of design operations driven by the same goal (Martinec et al., 2017).

Communication is at the core of the design activity; designers communicate their ideas using different means and representations. Due to a better understanding of the problem and solution added to the social interactions, their ideas evolve, and the designers' documents evolve accordingly, adding uncertainty and the need for more exploration and hence more iteration (Piccolo et al., 2019; Robinson, 2010). The large body of literature that analyses information behaviours of engineering designers asserts that designers expend 12% of their working time obtaining or receiving information, 8 %providing information, and 4% is dedicated to overhead activities associated with information transfer (Robinson, 2010). Active information gathering is the acquisition of precise information or user-specific information. In contrast, passive patterns are the reception of the information, whether requested or not.

Too much information can be as detrimental as too little, and optimum level is most beneficial. Empirical evidence suggests that moderate levels of communication lead to the most effective performance in engineering teams (Patrashkova-Volzdoska, McComb, Green, & Compton, 2003). Based on the literature, this research identified two states of communication between engineers and engineering teams. The state of gathering information is where the engineering designer actively seeks information, asking questions, researching or requesting information from external or non-human sources (Robinson, 2010). The state of disseminating knowledge is when the designer provides answers or shares information with the team in response

to a requirement. This state includes the response rate (Crowder et al., 2012) that informs the frequency and speed of answering a question.

This study has established that design processes can be either observed using different granularity levels, perceived as a complex system representing the overall processes or observed through the necessary steps performed during the design activity at the micro-steps of thinking during design (Lindemann, 2014). On a different note, Martinec et al. (2017) suggest that; to achieve enough abstraction for the domain and development context-independent analysis of the design operations, the identification of the elementary design operations as analysis, synthesis and evaluation is necessary (Cross, 2001; Liu and Lu, 2014). In this context, the study of the problem and its interrelations as part of the potential solution corresponds to the analysis stage (Jin and Benami, 2010), and the synthesis stage corresponds to the generation of the design solution. Moreover, the evaluation stage assesses the solutions' validity (Afacan and Demirkan, 2011).

Martinec et al. (2017) performed a protocolar empirical analysis of the design teams' activity, codifying seven sub-activities within the significant stages of analysis, synthesis, and evaluation, such as problem analysis, solution generation, solution analysis, evaluation planning and others. Their study found variations in the research, synthesis, and evaluation design operation sequences. Their protocolar analysis shows that an average proportion of the teams expended approximately 38% of the time in analysis, 54% in synthesis and 8% in evaluation; the study also revealed that patterns of transitions between operations are significant.

In this study, the design task considers the design operation sequences of analysis, synthesis, and evaluation; In the model, the analysis stage uses 30% of the time established for the task, the analysis stage uses 50% and synthesis 20%. Within the context of the stage gates model process, two types of iterations have been identified:

1. In-stage iterations, which happen within the stage, between each gate, with low or zero impact on previous decisions.
2. Cross-gate iterations, which happen after a decision, and after the next gate starts, affecting previous decisions that significantly impact the stage's progression.

In-stage iterations must be understood as something other than a repetition of the same activity. These iterations are learning cycles of validation and systems integration under realistic boundary conditions (Meboldt et al., 2013), seeking the overall system's improvement, and finding the best solution. Cross-gate iterations

significantly impact time and cost outcomes and must be prevented (Meboldt et al., 2013). However, the team must look for critical issues and assess them as early as possible.

The feedback loops perspective perceives the world as an interconnected set of circular relationships. Feedback is the transmission and return of information, informing the systems how they are doing concerning the desired state in a system's interrelated and mutually interacting components. Feedback loops represent the social reality of the system with negative and positive circles of causality. Feedback loops provide the continuous information necessary to bring the system under control (Chirumalla, 2017).

Time to the market or development time is necessary for elaborating simulation models. At the same time, several research studies have looked at the factors that influence changes in new product development (NPD) cycle time. A few companies have disclosed cycle times for specific projects; there is a need for more data on how long NPD takes.

The New Product Development Best Practices study (Griffin, 2002) asserts that industrial organisations take an average of 2.25 years (27 months) to create more innovative projects. Companies that incorporate all nine stages in their product development processes take an average of five additional months to complete. Furthermore, the stage of development (turning an idea into a functional prototype) takes an average of 8 months to complete. It is the most time-consuming step of the process; while testing and validation are 4.8 months, commercialisation or launch takes 4.5 months. Time spent in New Product Development is about 80.1% and occurs once the business case has been approved. However, almost all the empirical findings on project strategy are predictable. Longer development timeframes or longer time-to-market deliveries are related to newer, larger, more complicated, more technically demanding, and more creative projects. In contrast, shorter time to the market deliveries is related to projects with incremental innovations.

The information flows, deliverables, specifications, and other sources between teams allow the modelling of workflows and environments that reflect different types of product development projects organisations: the project characteristics, activities, the complexity of the organisation and the product being designed. Including innovation level and resources lead to the generation of the workflow, affecting its structure, mainly in activity dependence and iteration (Perišić et al., 2016). Activity

fragmentation should allow for the simulation of only specific activities or phases (Perišić et al., 2016).

The previous analysis allowed the identification of what and who is in the model. The aim of the model formalisation stage is to establish who does and when. Supported in the previous analysis and previous stages, the model narrative is developed as "an account" of series of events, facts, etc., given in order and with the established connection between them, table 4.4.1.1 presents the developed model narrative.

Table 4.4.1.1 Model narrative.

The agent types are the designers or design teams for each part defined within the system boundaries. The design task activities, analysis, synthesis, and evaluation are agent states performed/adopted by designers; those states change chronologically.

Communication activities are behaviours adopted by the agents during the process that influence the design task activity. The request for information is triggered in any design task stages (Dx), the agent will leave the design task execution, moving to communication states (Ix+Cx). From there, the designer/agent waits for an answer or answers a question. When the designer receives an answer, it moves from the state of gathering information to the state of evaluating the information (Ix). In the states of information evaluation, the agent decides to ask again, to carry on without the correct or complete information, or return to the design task when the information is complete (Dx). Feedback loops iterations manifest during the exchange information cycles between the designer, asking and answering and asking again; however, when the agent/designer decides to carry on without proper information, a cycle or rework is generated and has a direct impact on the workflow.

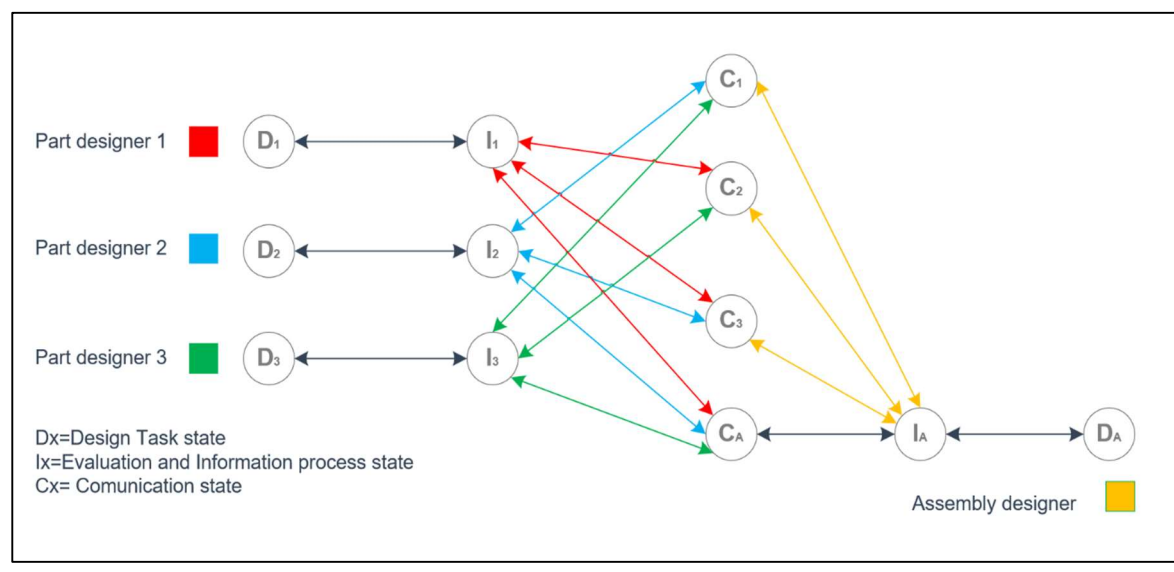


Figure 4.4.1.1 Communication patterns between the actors in model narrative.

4.5 Logical model

The previous stages have identified the system, model elements, relations, interrelations, agent types, behaviours, and states. At this stage, it is necessary to ensure the identified concepts can be translated into a computer language retaining their original meaning.

This research used abstraction levels or views of the system identified during the building intervention-evaluation artifact cycles in Section 3.2.2 of Chapter 3. These abstraction levels were used to model the tactical-level interactions of product development processes, including the product architecture representing the parts to be designed and the development process representing the process workflow. The fine detail operational levels at the microscale were used to model designer interactions such as design task that includes the methods of analysis and synthesis, and evaluation and the design activities, including communication and information processes.

The conceptual framework for simulating feedback loops in engineering design, is shown in Figure 4.5.1.1, and presents the individual parts process on the left side and the assembly or integration processes on the right side. In this process, the designer or design team needs to have the complete set of parts to proceed to the integration process; In this case, the workflow diagram shows the three queues where the parts must wait until all of them arrive, and the process can finally start.

The conceptual framework is implemented by grouping the system elements in four rectangular areas; (1) the design task includes the respective activities of analysis synthesis and evaluation, followed by the corresponding area of (2) the process and evaluation of the information that includes the "not communicated and effectively communicate activities", the next rectangular area include, (3) the gathering of information process with the correspondent "ask, ask again and answering".

Moreover, the area that includes (4) the development process is at the bottom of the diagram. The necessary designers or design teams for each part are represented as layers labelled as parts "a", "b", and "c". The model uses circles to depict the actions or activities of the designer within the *design task* and *design activities*. In the

process development, rectangular shapes represent blocks to illustrate the meso-level operations.

This thesis has established that interaction points are pairs of information exchange between the models that are correctly “captured by” or “influenced by”. The simulation modelling approach is the result of mapping the boundaries of the models that need communication. The framework identified five preliminary interaction points between the meso-level and the micro-level models. The first interaction point releases the design requirement into the workflow, which triggers the process start in the design task. Second, the design process time is governed by the design task activity and directly influences the process block in the workflow. The third interaction point generated in the evaluating information activity influences the decision block by sending back the processed design requirement. The fourth interaction point also affects the decision block; the carry-on block influences it, and the effect is to send the already processed design requirement to a rework queue. Finally, the fifth interaction point passes the approved design requirement to go to the integration model.

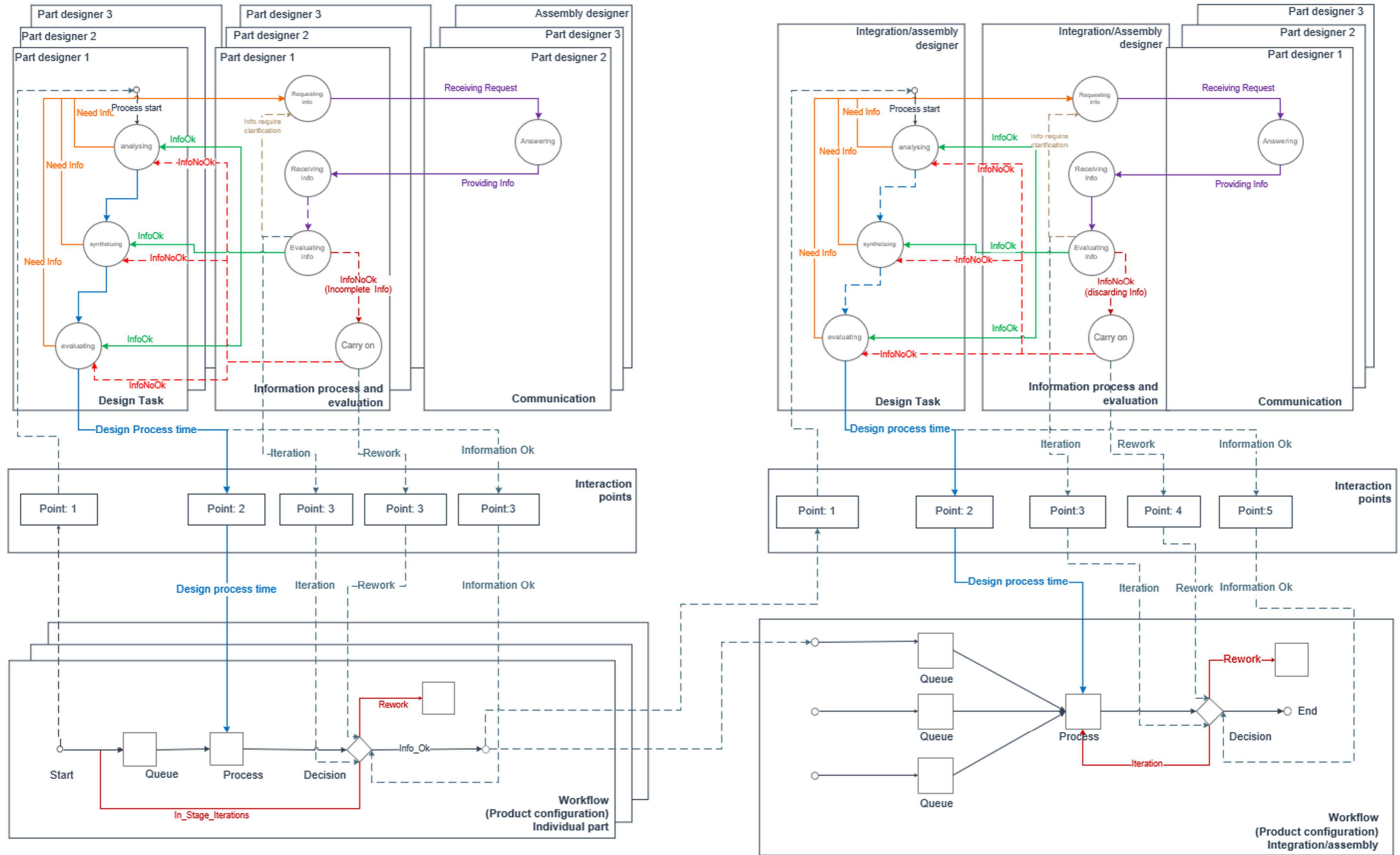


Figure 4.5.1.1 Conceptual framework for simulating feedback loops in engineering design.

4.6 Summary.

This thesis stands for the understanding that conceptual modelling activity aims to clarify the meaning of ambiguous terms and avoid problems with different interpretations focusing on capturing the representation of human perceptions of the natural world to be included in an information system (Pastor, 2016) that can be specified using verbal narratives and sketches (Grimm et al., 2020) allowing the development of a conceptual framework for simulating feedback loops in engineering design presented in the previous sections. For developing a conceptual framework for simulating feedback loops, this study followed a process that includes five main stages: (1) Problem identification, (2) system identification, (3) system (4) conceptualisation, (4) model formalisation and (5) Logical model.

The problem identification stage, through literature analysis, found that the product development system, which receives inputs in the form of market opportunities from the environment, is significantly impacted by social-organizational aspects. These aspects include factors such as communication and coordination mechanisms and social and cultural characteristics, such as individuals' background and level of experience. The product development is also shaped by the process characteristics, which are defined as the set of activities performed to execute the product development project. This includes hardware and software modules and components that are specified in the product architecture design to produce the intended product or service. This thesis also established in the conceptualisation of the research opportunity in the problem formulation Section 3.2.1 of the ADR method in Chapter 3, that product development systems are a multidimensional network of intricately linked operations with feedback loops across several levels of hierarchy, and these feedback loops increase complexity, time to market, and cost and are essential elements of engineering design processes. Feedback loops have a beneficial effect on design outcomes (i.e., the quality of the final design) (Tapia et al., 2021); nevertheless, while loops arising from rework have a positive effect on the final design, their negative influence on project development outcomes may be significant.

System identification consists of an inventory of the system components, explicitly identifying relevant concepts, actors and objects, appropriate behaviours, interactions or flows, states, or properties (Nikolic and Ghorbani, 2011). From the problem formulation, the relevant system elements identified are actors, who are the designers or design teams for components. Actors exhibit behaviours of asking questions, answering questions, asking again, and can carry on without proper

information and performing a non-coordinated rework. Actors perform design activities triggered by a design brief or receiving or requesting design information. Interactions then lead to design iterations and feedback loops triggered between actors.

In the system conceptualisation, the system under analysis must be explicit, formal, and expressed in a way to be understood by computers and human beings (Nikolic and Lukszo, 2013). This thesis uses ontologies to help in the process of formalisation of concepts and relationships without focusing on software implementations. In the ontology, “what” is the model, including all entities within the system boundaries and its relationships are “formally encoded”. Even though there “is no single correct ontology design methodology” (Noy and McGuinness, 2001).

The developed ontology model allowed the concepts' formalisation to be used during the simulation model construction in the implementation stage. The ontology developed using Ottes' (2019) approach enabled the formalisation of the interactions between the system elements, formalising the relationship between processes, information and material entities.

In the model formalisation stage, two main tasks were performed: first, creating a model narrative as an account of a series of events and facts, given in order and with an established connection between them (Nikolic and Lukszo, 2013). That includes an analysis of two relevant frameworks for the simulations of engineering teams to identify those variables useful in developing the narrative in the simulations of engineering design teams.

The logical model implementation consisted of a graphical representation of the conceptual framework implemented, using the abstraction levels or views of the system identified during the building intervention-evaluation artifact cycles. These then model the tactical-level interactions of product development processes, including the product architecture representing the parts to be designed and the development process representing the process workflow.

The conceptual framework is implemented by grouping the system elements in four rectangular areas; (1) the design task, followed by the corresponding area of (2) the process and evaluation of the information, (3) the gathering of information process and the (4) the development process workflows at the bottom of the diagram. The framework identified five preliminary interaction points between the meso-level and the micro-level models.

The major challenge for the practical analysis of product development processes that this conceptual model addressed are (1) the comprehension of process behaviour

and (2) the understanding of the characteristics and behaviours of engineering designers' communication patterns. The analysis of how human and organisational feedback loops affect design task performance in engineering design processes, either positively or negatively. Is a key contribution. The technical systems in the form of product development processes and product architecture, were also included as relevant features for the simulation feedback loops in engineering design.

4.6.1 Agile Scrum Methods

The agile scrum methodology is defined as a method to organise and follow up on the design tasks and activities of design teams, suggesting a control on design requirements and progression of the development tasks dividing the project into small design projects and controlling progression through what is called a sprint backlog. In the agile sprint method, the iterations are incremental cycles that contribute to the progressive completion of the tasks and improve quality.

Within the classification of the iterative approaches proposed in section 2.5.1, sprints can be categorized as strategic iterations, where is expected that appropriate strategies and policies influence the process and the final designed product.

A sprint includes sprint planning, sprint execution and sprint retrospective, And t the work is regarded as being complete when it is deemed to be a good quality and, a potentially, shippable product.

Sprint Iterations are governed by the sprint backlog and assessed among the criteria of the potentially shippable product, which is defined by the Product owner and the sprint development team, based on the customer requirements. In the scrum sprint methodology, when the sprint team do not reach the potentially shippable product, the incomplete sprint is reorganized in the sprint backlog and needs to be revisited in the following sprint which could be recognised as rework.

Cooper and Sommer (2018) assert that when a scrum is implemented within a stage gated process, gates remain an important part of the hybrid model, providing vital go/kill decision points, enabling management to review projects at key transition points (Cooper and Sommer, 2018).

Mvulane (2020) explains that the Scrum methodology comprises key elements, including a product owner, Scrum master, and a development team. It is integrated with various components such as sprint planning, daily stand-up meetings, development (sprint) iterations, sprint reviews, and sprint retrospectives. The Scrum process involves essential components: Scrum roles, Scrum events, and Scrum artifacts.

The Scrum roles include the product owner, the Scrum master, and the development team. The product owner's primary responsibility is to maximize the value of the product, estimate the budget (Mvulane, 2020), and work closely with stakeholders. This role involves managing the product backlog by collecting stakeholders' requirements, defining, and prioritizing tasks. The product owner plays a crucial role in deciding the prioritization of backlog items and addressing the consequences of those decisions (Sif, Thor, & Ingi, 2014). It is vital for the product owner to maintain a balance between the different product features when prioritizing items (Mvulane, 2020).

The Scrum master's primary function is to ensure that the Scrum team adheres to the theory, practices, and rules of the Scrum methodology, thereby ensuring a deep understanding and enactment of Scrum philosophy (Rossberg, 2019). The Scrum master is also responsible for removing impediments and creating a conducive working environment where the team can work at a sustainable pace (Rossberg, 2019).

The development team works on completing the sprint tasks specified in the product backlog. They are responsible for producing increments during the development sprint iterations and ensuring that these increments meet the definition of done (Mvulane, 2020). The development team is also responsible for updating the Scrum board during daily stand-up meetings and managing the sprint backlog.

The Scrum artifacts encompass the product backlog, sprint backlog, and Scrum board or burndown chart. The product backlog consists of an ordered list of product requirements established by stakeholders in collaboration with the product owner. This list includes business requirements, technical and functional requirements, and

nonfunctional requirements, all of which are part of the definition of done (Rossberg, 2019).

The sprint backlog outlines what the development team needs to deliver during the sprint and includes a plan for how the team will work to achieve the product increment. Tasks in the sprint backlog are broken down during the sprint planning. The development team has ownership of the sprint backlog and can add or remove tasks as needed (Rossberg, 2019). The team tracks their progress using a burndown chart (Scrum board).

The sprint planning meeting is used to discuss changes in the product backlog. This meeting is timeboxed to eight hours for a calendar month. During these meetings, the team makes decisions about what and how the increment will be delivered. The key Scrum events are Sprint, daily scrum, sprint review, and sprint retrospective.

The daily scrum is a daily meeting where the team reviews what was completed in the previous sprint, discusses the goals for the day, and identifies any restrictions or risks that may impede progress. The sprint review takes place at the end of the sprint iteration, during which the team showcases what has been accomplished to the product owner and stakeholders. The suggested time for this meeting is three hours for a calendar month sprint, and the outcomes of the sprint review inform updates to the product backlog or the retrospective following the review (Rossberg, 2019).

The sprint retrospective is used to agree on continuous process improvement actions, and the duration of these meetings should be one and a half hours for a two-week sprint. Rossberg also defines backlog refinement as an ongoing process to review the product backlog items and appropriate prioritization, breaking down larger product backlog items into smaller ones to fit them into the sprint planning. Within the backlog refinement, the acceptance criteria are clarified (Rossberg, 2019).

The definition of done outlines the requirements, including the stakeholder's acceptance criteria, which need to be fulfilled. It is a primary quality document that defines what constitutes a finished stage of the product to be delivered as a product increment. A product increment is understood as the sum of all backlog items completed during a sprint, including items delivered from previous sprints. At the end of each new sprint, the new increment must meet the requirements outlined in the definition of done.

The following table summarizes the key elements of the scrum sprint methodology.

Table 4.6.1 Summary of the Scrum sprint events, roles and artefacts based on Mvulane, 2020).

			Sprint iteration events				
Categories	System elements	Functions	Sprint planning	Daily standup meetings	Sprint	Sprint Review	Sprint retrospective
Scrum Roles	Product Owner	Oversees the development, defines the sprint scope and vision	Define priorities in product backlog and define sprint scope			Asses the increment among the stablished goals and reviews the definition of done	Ensures that the team inspects the increment and makes improvements
	Scrum Master	Is in charge of the maximization of the sprint effort and to protect the team from distractions.	Interacts with e product owner and manages the challenges of the sprint backlog	Ensures the team attends to the meetings	Protects the team from distractions during the sprint	Ensures the meeting schedule and the participation of the stakeholder	Facilitates the inspection and adaptation to scrum by the development team
	Development Team	Develops the product, based on the defined requirements of product backlog and performing the tasks defined in the sprint backlog	Define sprint goals with the product owner and priorities in the sprint backlog	Updates the scrum board and inspects the progress towards to completed work	Design, test and develops the product as per the requirements stablished in the sprint backlog	Shows the product increment to the stakeholders, the scrum master and the product owner and compares among the sprint backlog	Inspects the product increment, and reviews suggest increments to the sprint process
	Stakeholder	Define requirements and the acceptance criteria				Provides feedback and agrees with the product increment presented	
			Product backlog				
			Sprint backlog				
			Scrum board				
			Scrum artefacts				

Because the conceptual framework for the simulation of feedback loops in engineering design takes a socio-technical approach, it considers technical aspects such as technology, infrastructure, and processes, alongside socio aspects like people, goals, and culture. This approach recognizes the systematic connections between these elements (Clegg et al., 2017). The study suggests that new product development processes within design domains, like the Scrum sprint agile methods, which involve team collaboration, complex problem-solving, creativity, and information exchange, are representative examples of complex socio-technical systems (Robinson, 2016).

The conceptual framework for simulating feedback loops in engineering design has four stages for analysing and implementing a simulation model: conceptual model development, simulation model implementation, verification and validation, and documentation.

The development of the conceptual model involves five main stages: (1) problem identification, (2) system identification, (3) system conceptualization, (4) model formalization, and (5) logical model.

For problem identification within the conceptual framework, it is essential to establish and understand the problem that needs to be addressed. The "problem" is viewed as a poorly understood situation that exhibits suboptimal performance or signifies a lack of knowledge about a system, its behaviour, or its response to interventions (Nikolic and Lukszo, 2013).

4.6.1.1 Problem Identification in Scrum

Scrum is an iterative and incremental process where development activities occur in timeboxed intervals known as iterations, typically lasting from two to four weeks. During iterations, daily scrum meetings are conducted to inspect the work done, enabling necessary adaptations and adjustments to the design to be quickly identified and implemented. Within the Scrum methodology, the "definition of done" determines the completion of the design activity within the established stages (Rossberg, 2019). This definition outlines what user requirements, in addition to the user's acceptance criteria, must be fulfilled. It serves as a preliminary quality document.

Iterations conclude with inspections, and this cycle continues until the project is no longer funded (Rossberg, 2019). The key challenge in the Scrum sprint method lies in achieving a balance between what is considered within the definition of done and the sprint backlog activities performed before the project's funding ceases.

4.6.1.2 System Identification

The criteria for identifying system elements in the framework for simulating feedback loops in engineering design suggest that actors are entities capable of decision-making, while all others are considered objects. Properties that describe and specify agents are called states. Interactions refer to in-out interplays between agents, and behaviors are state changes resulting from interactions (Nikolic and Lukszo, 2013).

Agents are the fundamental units of the model, representing one or more actors within the system. They are distinguished by their boundaries, states, behaviors, and

interaction capabilities. Interactions can vary in terms of duration, immediacy, and impact.

For identifying system elements in the Scrum sprint, the product owner, scrum master, development team, and stakeholders can be considered as actors. Sprint iteration events can be identified as relevant behaviours and interactions, while sprint development is regarded as a state or property. Within the category of Scrum artifacts, product backlog and sprint backlog can be seen as interactions, while the Scrum board is an object with no interactivity with the agents. Table 4.6.2 summarizes the identification of system elements and their categorization using the system identification criteria proposed in the conceptual framework for simulating feedback loops in engineering design.

4.6.1.3 System Conceptualization

The previous stage provided a general identification of the system elements in the Scrum sprint and a preliminary categorization of interactions, agents, states, and behaviours, resulting in an inventory of system components. However, the identified concepts expressed in natural language are unsuitable for computer interpretation. The model of the world needs to be explicit, formal, and comprehensible to both computers and humans.

The formalization activity aims to generalize the system description beyond a single domain. Ontologies can assist in the formalization of concepts and relationships without a focus on software implementations. Two approaches to ontology implementation are used in this thesis. In the first approach, an instance represents a single identifiable object within the model's scope, and a class is a generalization of several instances.

In the second approach, Otte et al.'s (2019) suite of modular ontologies for the product life cycle and its successive phases is employed. This modular ontology identifies material entities, information entities, and processes. It captures relationships with a two-place instance-level relationship, involving artifacts and human agents. The terms defined by Otte et al. (2019) are used to identify relationships between the system elements, specifying that material entities are agents with a physical existence, processes relate to planned or unplanned activities required to achieve a goal, and information entities represent various forms of communication, related or unrelated to processes or material entities.

The systems conceptualization stage concludes with the implementation of the system ontology, as presented in the following diagram. The development of this ontology facilitates the formalization of interactions between system elements

concerning processes, information, and material entities, following the criteria established in Section 4.3.

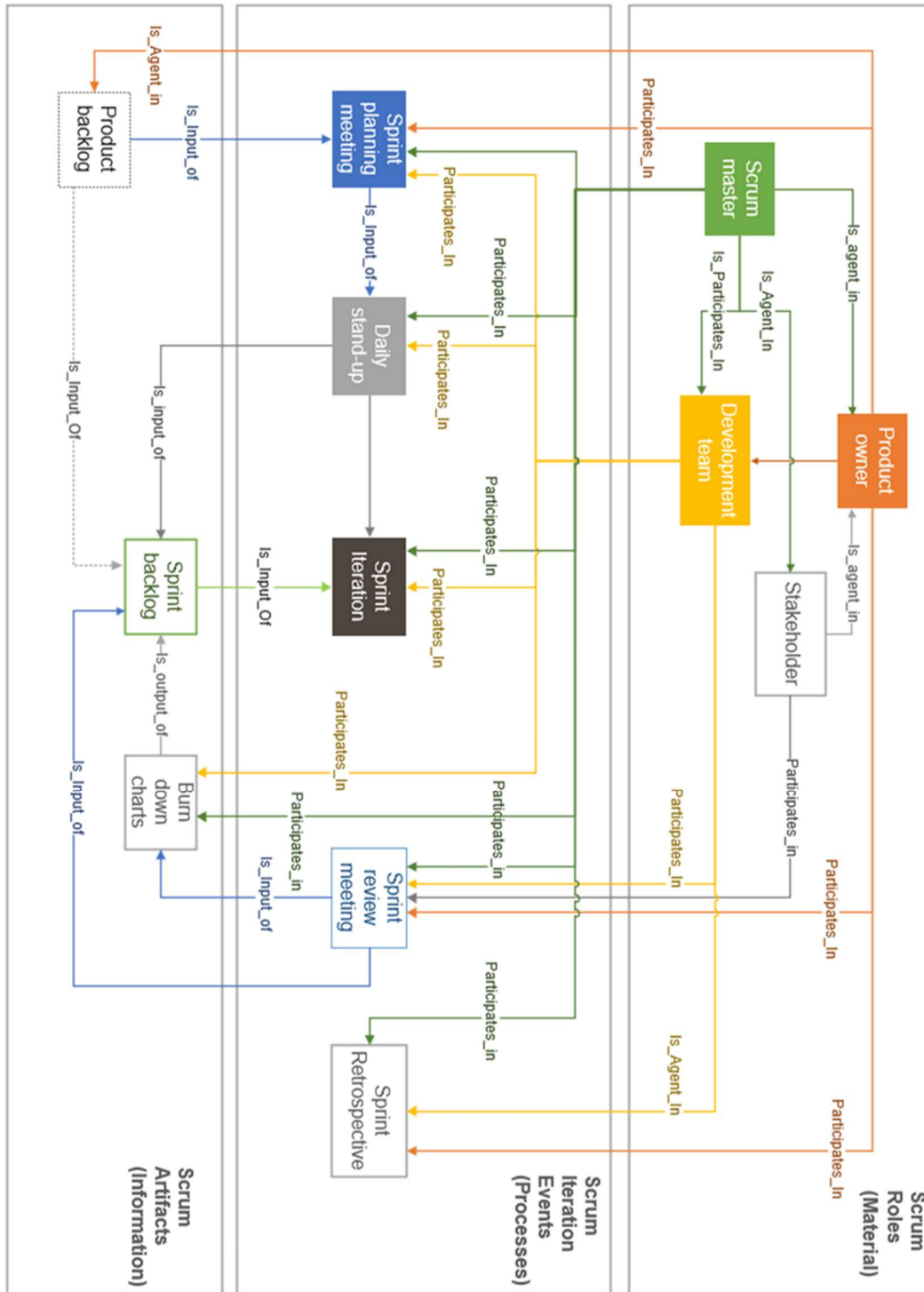


Figure.4.6.1.3.1 Scrum sprint ontology.

4.6.1.4 Model Formalization

The preceding stage has identified who and what is within the model. In the formalization stage, the objective is to define who performs what actions and when. This stage seeks to transform the previously identified concepts into an artificial model with an abstract representation, creating a model narrative that presents a series of events and facts in a sequential order with established connections between them.

The system elements of the traditional product development process identified in Section 4.3 of the model conceptualization stage can be equated with the system elements of the Scrum sprint methodology. In Table 4.6.1.4.1 Traditional product development process versus scrum sprint method, the left side comprises the system elements of the traditional product development process, while the right side encompasses the system elements of the Scrum sprint methodology. The classification of Otte's ontology, including Material, Processes, and Information entities, aligns with the Roles, Events, and Artifacts of the Scrum method.

Table 4.6.1.4.1 Traditional product development process vs scrum sprint method.

Traditional product development process					Scrum sprint methodology	
System elements	Material	Process	Information	Otte's ontology classification	Categories	System elements
Part	✓			Material entities	Scrum Roles	Product Owner
Designers	✓					Scrum Master
Product development		✓		Process entities	Sprint iteration events	Sprint planning
Communication process		✓				Daily standup meetings
Design Task			✓			Sprint
Design activity		✓				Sprint Review
Processing information		✓				Sprint retrospective
Feedback loops			✓	Information entities	Scrum artefacts	Product backlog
Design brief			✓			Sprint backlog & Scrum board
Product architecture			✓			Definition of done
Evaluating information		✓				

Material entities are agents or elements from the real world. These correspond to the Scrum roles, where the Product Owner, the Scrum Master, and the Development Team are individuals involved in the process, each performing distinct activities. In

the traditional product development process, these roles are akin to designers and design teams.

Processes are associated with planned or unplanned activities, representing the actions required to achieve a goal. In the context of Scrum, these are reflected in the sprint iteration events, which are carefully planned activities with specific timeframes and durations. The sprint iteration aligns with the design task, wherein designers conceptualize, design, and evaluate solutions, and with design activities that encompass actions beyond the design task, such as communication and information behaviours of the designers while executing the design tasks.

Scrum artifacts are the means through which designers communicate their ideas and control the progression of the design process. The design brief corresponds to the product backlog, capturing the requirements of the stakeholders. The product architecture informs the configuration of the product development, like how the sprint backlog captures and prioritizes the necessary tasks for the design process.

The previous analysis conducted during systems identification and systems conceptualization suggests that the identified variables and characteristics defined as elements of the scrum sprint methodology are valuable for the application of the conceptual framework for the simulation of feedback loops in engineering design, facilitating the simulation of the scrum sprint processes.

4.6.1.5 Logical model

Having identified what and who is in the model. The next step in the formalisation stage is to establish who does what and when. The model narrative is developed as an account of series of events, facts, etc., given in order and with the established connection between them. The narrative in Figure 4.6.1.5.1 describes how the sprint process develops.

In previous stages, the system, model elements, relations, interrelations, agent types, behaviours, and states have been identified, this stage, aims to ensure the concepts can be translated into a computer language retaining their original meaning. Table 4.6.1.5.1 present the possible variable identified.

Table 4.6.1.5.1 Variables identification.

Categories	System elements	Task	Info direction	Recipient status
Scrum Roles	Product Owner	Define sprint scope and vision	Out	Active
	Scrum Master	Protects the team	In/Out	Active
	Development Team	Performs product design	In/Out	Active
Sprint iteration events	Sprint planning	Is used to define priorities	In/Out	Active
	Daily standup meetings	Updates scrum board and inspects progress	In/Out	Active
	Sprint	Design and development activities	In/Out	Active
	Sprint Review	Is used to show and asses product increment	In/Out	Active
	Sprint retrospective	Is used to provide process feedback and to inspect the product increment	Out	Active
Scrum artefacts	Product backlog	Is used to capture product requirements	In/Out	Passive
	Sprint backlog & Scrum board	Is used to organize and prioritize tasks and activities	In/Out	Passive
	Definition of done	Is used to determine what the acceptance criteria in each stage	Out	Passive

Sprint Narrative

The implementation of Scrum typically begins with a client's request for product development (Streule, Miserini, Bartlomé, Klippel, & De Soto, 2016). Upon receiving the client's request, the product owner proceeds to prioritize items within the product backlog. These items are then further broken down into tasks during the Sprint Planning meeting. During this meeting, the development team estimates the duration of the sprint based on the tasks recorded in the sprint backlog. They break down what the team needs to do to deliver "the increment" and estimate the hours required to complete each task, deducing the number of tasks to be completed in the sprint.

A sprint is a time-bound event (Dalton, 2019) during which product development activities occur. The development team sets a time interval for completing the Sprint iteration, which comprises four main phases, (1) Sprint Planning, this meeting, as described earlier, is where the development team breaks down the items recorded in the product backlog into manageable tasks and estimates the number of hours required for each task. (2) Daily Stand-up meetings, these brief meetings are used to track the status of tasks and to report progress towards meeting the sprint goal (Rossberg, 2019). They are typically held for 15 minutes every day at the same time and place.

(3) Sprint Review, this meeting is used to discuss any shortfalls encountered and to inspect the results, assessing the products against sprint goals and requirements. The sprint review typically lasts about four hours after the conclusion of the sprint. During the Sprint Review meeting, if all tasks in the sprint backlog are completed, the increment is then handed over to the product owner. However, if not, the whole process begins again from the sprint-planning phase. (4) Sprint Retrospective, this meeting typically lasts for about three hours, during which the team inspects, adapts, and improves the work procedures in preparation for the next sprint.

In Scrum, each sprint is created to develop a product, meant to produce increments for a larger product (Bibik, 2018). After the completion of the sprint, an increment is produced by the team only if it meets the requirements established in the definition of done (Rossberg, 2019).

Figure 4.6.1.5.1 Sprint process narrative, based on Mvulane, (2020).

The conceptual framework has been integrated with Scrum-sprint constructs to facilitate the construction of the simulation model. This conceptual framework utilizes three rectangles that represent the original three components: process, information processing and evaluation, and communication. It then allocates the Sprint agents to their corresponding components. The Development Team corresponds to the block

of design activities, the Scrum Master to the information processing and evaluation, and the Product Owner is assigned to the communication block.

The framework proposes three initial interaction points: "Process start," which injects agents from the workflow to the agents; from the Sprint Review to the process, controlling the flow; and a trigger events relationship from the "Definition of Done." According to the definition of done, this relationship determines what is considered completed. This application illustrates the flexibility of the proposed framework, as it analyses and implements design processes from a socio-technical perspective, encompassing people, processes, and technology.

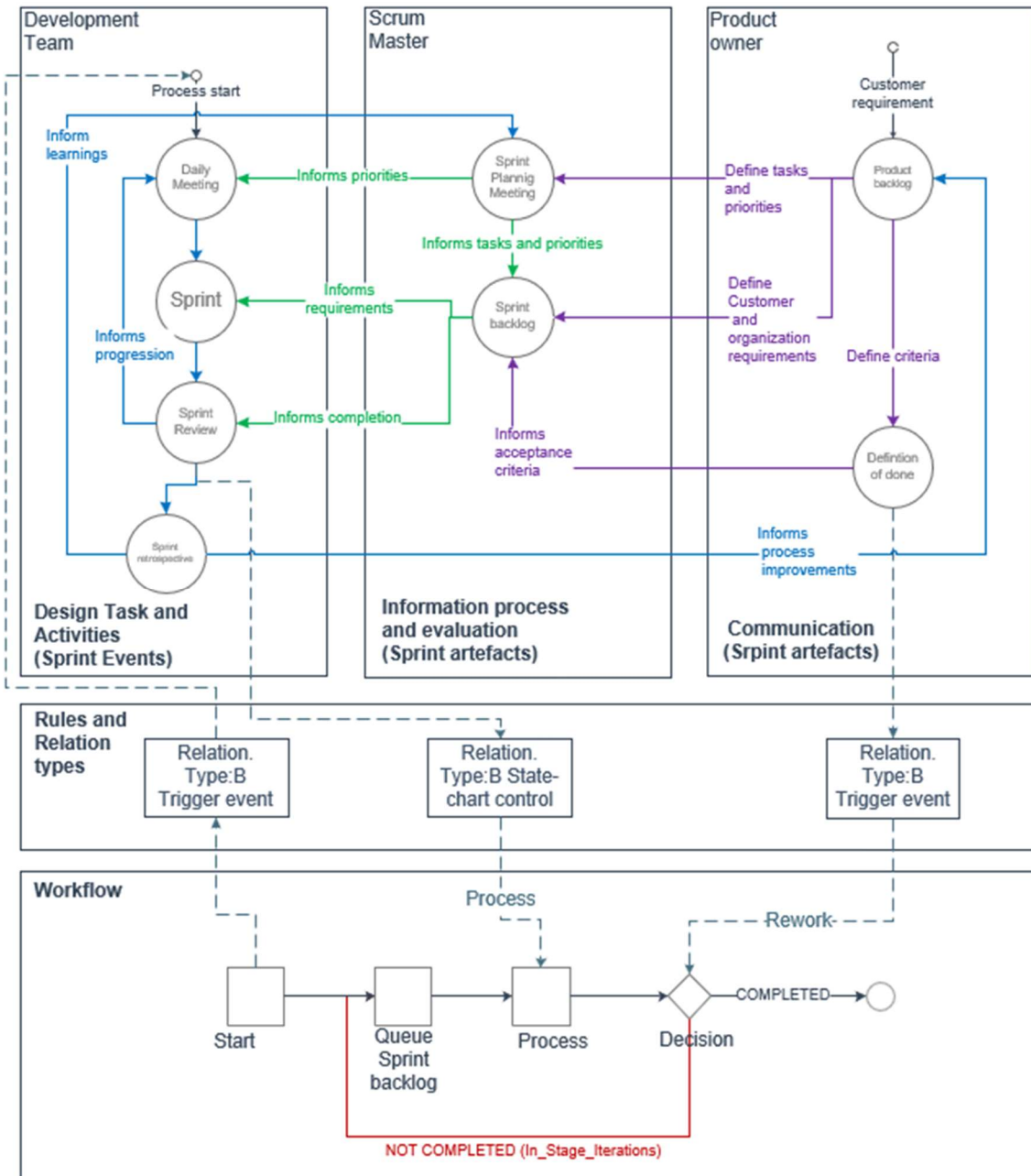


Figure.4.6.1.5.3 Conceptual framework adaptation for the simulation of agile sprint processes.

In summary the previous implementation of the conceptual model demonstrates that the conceptual framework, originally designed for simulating feedback loops in engineering design, possesses the flexibility to be adapted for simulating Agile sprints. This adaptation considers the socio-technical characteristics inherent to agile processes. By repopulating the conceptual framework with Scrum-sprint constructs, the model successfully integrates the elements, relations, agent types, behaviours, and states specific to agile methodologies.

Notably, the model allocates Scrum agents to their corresponding components: the Development Team, Scrum Master,

Chapter 5: Conceptual framework implementation

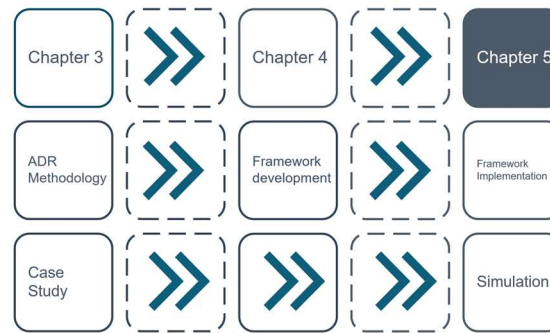


Figure 5.1 Chapter layout.

This chapter introduces the simulation model implemented after the conceptual framework phase in Chapter 4 using the case bicycle study introduced Section 3.4.

This thesis has examined the Mykoniatis and Angelopoulou's (2020) framework for the integration of different simulation methods in other domains as socio-technical, cyber-physical, business systems, and healthcare organisation. This framework provides a structure for the implementation of multiparadigm simulation models, integrating agent-based, discrete events, and system dynamics simulation methods. Mykoniatis and Angelopoulou (2020) identified three critical questions to guide the implementation of the simulation approach; The first question attempts to establish why and when a real-world system requires multiparadigm modelling and simulation.

The second question concerns the interaction points among the simulation models used. The third question asks how the simulation models interact with each other. To answer the first question, this thesis has established that realistic simulations with a comprehensive view of the engineering design process must consider the technical aspects of the product development, such as the architecture of the product, the linear logic of the development process, along with the social aspects of design teams' communication patterns and feedback loops.

Those considerations must include the identification of the organisational activities in engineering design and development processes, such as those related to individual actions and their context (micro-level), those associated with the flow of tasks and design progression (meso-level), and those related to project/programme and contextual considerations (the macro-level), as part of the complex dependency structures of the product development processes (Wynn and Maier, 2022).

5.1. Implementation

The problem and its underlying system were established in the previous chapter during the conceptualisation stage in Chapter 4 of this thesis. This stage involves defining the details necessary for programming the simulation. During this process, changes to concepts and the model design may occur during implementation due to unforeseen outcomes or issues (Nikolic and Ghorbani, 2011).

The development of the conceptual framework for simulating feedback loops includes five main stages, (1) problem identification, (2) system identification, (3) system conceptualisation, (4) Model formalization and (5) Logical model.

The problem identification defined in Chapter 4, Section 4.1.1, established that product development processes within engineering design are significantly impacted by social-organizational aspects, including factors such as communication and coordination mechanisms, and social and cultural characteristics. Product development is also shaped by process characteristics, defined as the activities performed to execute the product development project. This includes hardware and software modules and components specified in the product architecture design to produce the intended product or service.

For the implementation of the simulations using the conceptual framework developed in the previous chapter. The handlebar assembly case study in this research is an example of an engineering design process, starting with identifying the design structure (shown in Figure 3.3.1.3). The bicycle handlebar assembly integrates the individual parts, brake lever, gear change, and handlebar. Each feature needs to be designed independently but integrated into the assembly at the end of the process.

Section 3.3 of Chapter 3 established that the design process begins with each team receiving a simultaneous design request, which initiates their respective design activities. However, the handlebar assembly design team must wait for the completion of all three component designs before proceeding. During the design process, iteration may occur within each design team during the processing and evaluating of the information used in the design task. Effective and timely communication is therefore essential to prevent rework and avoid a vicious cycle.

The systems identification stage in the conceptual framework (see Section 4.2) prescribes the elaboration of an inventory of the system components identifying relevant concepts, actors, objects, behaviours, interaction flows, states or properties. The outcome of the analysis of the identification of system elements is a list of 21 system components identified and classified is presented in Table 5.1.1.1, the table

of the system concepts and their respective classification show in the first column is the consecutive numbering and in the second is the system element name. In the following columns, the categories are (a) Concepts, (b) Actors, (c) Behaviours, (d) Interactions, and (e) States.

Table 5.1.1.1 Concepts, Actors, Behaviours, Interactions, and States identification.

No.	System elements	a) Concepts	b) Actors	c) Behaviours	d) Interactions	e) States
1	Bicycle design	✓				
2	Design request				✓	
3	Brake lever		✓			
4	Gear change		✓			
5	Handlebar		✓			
6	Handlebar assembly		✓			
7	Part design				✓	
8	Designer iterates				✓	
9	Communicates					✓
10	Asking questions			✓		
11	Answering questions			✓		
12	Feedback loops	✓				
13	Communicated effectively					✓
14	Not communicated					✓
15	Incomplete information					✓
16	Decide to ask			✓		
17	Carry on without proper information			✓		
18	Not coordinated			✓		
19	Rework				✓	
20	Not coordinated rework			✓		
21	Vicious circle	✓				

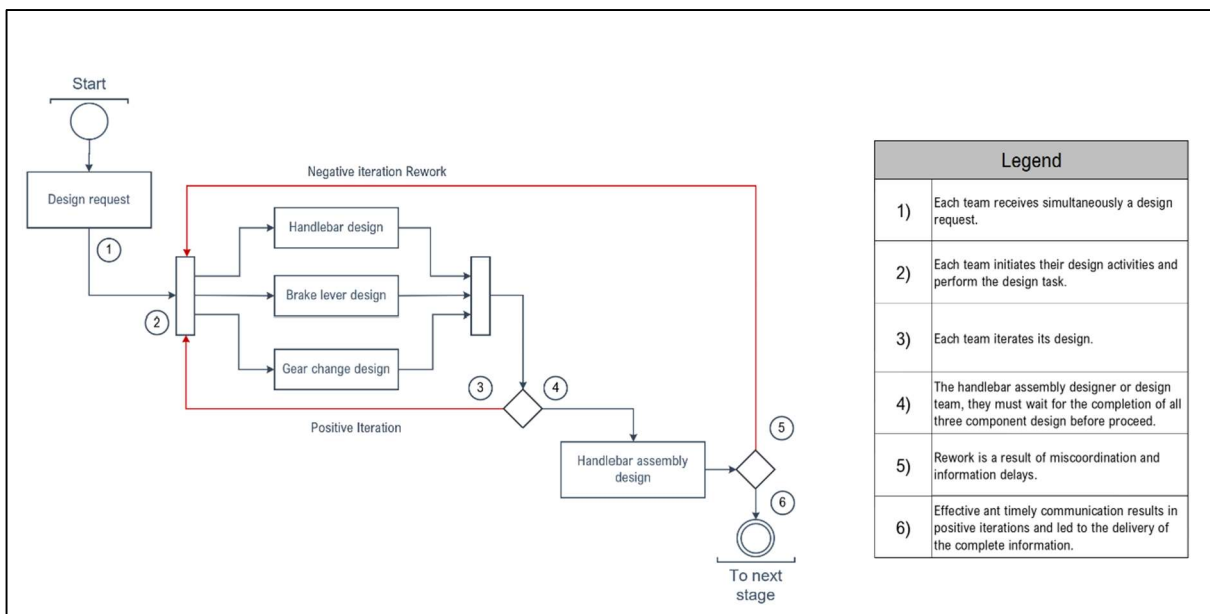
The system conceptualisation of the case study is supported by the process instrumented in Section 4.3 in Chapter 4 of this thesis. the relationships of four identified designers or design teams/actors necessary to perform the design of the handlebar assembly, their relationships, the corresponding interactions, the design task, and design activity, which includes information processing and evaluation and communication. The correspondent behaviours and the three states that can describe the agents in the handlebar design case study are also provided (1) the agent performs a design task, (2) process and evaluates information, and (3) communicates are shown in table 5.1.1.2.

Table 5.1.1.2. System components identification for the handlebar design structure.

Actors	Relationship	Interaction	Behaviours	States
Brake lever designer or design team	Sub-systems for the handlebar assembly	Design Task	Works in the design task	Analysis, Synthesis and Evaluation
Gear Change system designer or design team		Design activity/communication	Ask for information	Communicate
Handlebar designer or design team			Answers to information requests	
Handlebar assembly designer or design team		Design activity/information processing and evaluation	Determines the need for information	Evaluate information
	Provides information			
	Decide to carry on			

In this thesis, the model formalisation stage has been established in Section 4.4 that model narrative describes the events in the given order and the established connection between them. A model representing the narrative of the handlebar design process using Unified Modelling Language (UML) diagram is presented in Figure 5.1.1.1

Figure 5.1.1.1 UML activity diagram for the handlebar design process narrative.



5.1.2 Identification of MS methods

Section 3.2.2 of this research established the use of the Djanatliev and German (2015) method to guide the prototyping/development of domain-specific hybrid simulations. The framework provides a macro-structure for implementing simulation models with three critical processes for structuration of the simulation scope.

The framework asserts that identifying the abstraction levels is possible by dividing the overall domain scope into specific subclasses. Independent simulations can be implemented through a hierarchical breakdown where macro, meso, and micro levels cover the actual situations within the simulation context. The framework prescribes that explaining how simulation models are linked to the abstraction levels is essential to determine: the relevant continuous structures if they are present, if processes must be traversed, or if the representation of individual behaviours is necessary (Djanatliev & German, 2015).

The literature about the modelling of the iterative process suggests three approaches to identify a system's independent levels. (1) The information-based approach uses the information about the process to determine process behaviour and capture feedback associated with iteration. (2) Task-based approaches consider the design process as a collection of tasks attempted and revisited until completion. In this application, each iteration is considered the repetition of a completed task or the execution of the same task in a different context. (3) The actor-based approach is based on the modelling of multiple agents and their coordination process; in this approach, iteration is perceived as a function of the continuous dialogue between process participants (Wynn et al., 2007).

The modelling approaches identified by Wynn (2007) can be mapped to determine the abstraction levels suggested by Djanatliev and German (2013), considering the critical features of the design processes simulated in this research. The information-based approach can be identified as a high-level abstraction paradigm and the system's dynamics can be used at this level however is not used in this research. The tactical task-based approach, with medium detail and operational interactions in fine detail, can be used at the meso-level suggested by Djanatliev and German. Finally, the actor-based approaches, at a low abstraction level with detailed data, can provide the system's micro-level view. In this thesis has been established that, the stage-gates model provides the chronological structure (the process) and the product architecture to form the development process structure for designing a give product deriving the workflow. The discrete events simulation modelling captures the task's progression, the process structure, and the cross-gate iterations (rework).

Here, the stage-gates model provides the chronological structure (the process), and product architecture is used to identify the parts being designed to form the development process structure for designing a given product, deriving in the workflow. The engineering design processes identify systems and subsystems of the designed product and the social interactions of the designers (the people). The agent-based simulation modelling can capture the design team's micro-level interactions and information patterns, sharing and evaluating information while performing design tasks, and the iteration feedback loops of individual designers. On a different note, rework stage-gated feedback loops govern the development processes, therefore, are well-suited to discrete event simulation and on the other hand, design iteration feedback loops are driven by individual designers and best modelled using agent-based simulation (Tapia et al., 2021). Figure 5.1.2.1 illustrate the coupling of the abstraction levels (left side), simulation approaches (middle part) and the simulation methods used (right side).

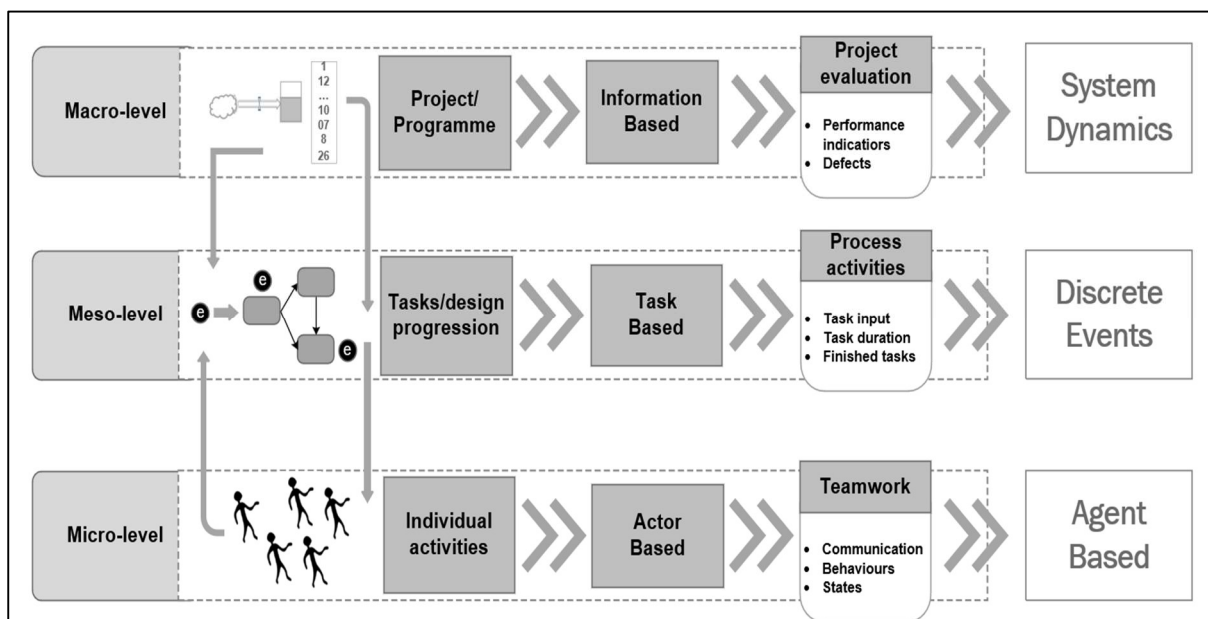


Figure 5.1.2.1 Identification of abstraction levels, simulation approaches and methods adapted from Djanatliev and German (2013).

5.1.3 Identification of the interaction points.

The stage of identifying the abstraction levels and the simulation approach has been completed by describing how to use the simulation paradigms to model structures at the considered level. The next step describes the interaction between abstraction

levels defining how the simulation-related elements can be connected (Djanatliev and German, 2015).

Interaction points are the pair of inputs-outputs of data exchange between interacting models that need to communicate, resulting from mapping the models' boundaries. According to the framework for the implementation of hybrid simulations proposed by Mykoniatis and Angelopoulou (2020), there are two kinds of inputs, two types of outputs and four boundaries to be considered during the process of the interaction points identifications as shown in the Figure 5.1.3.1

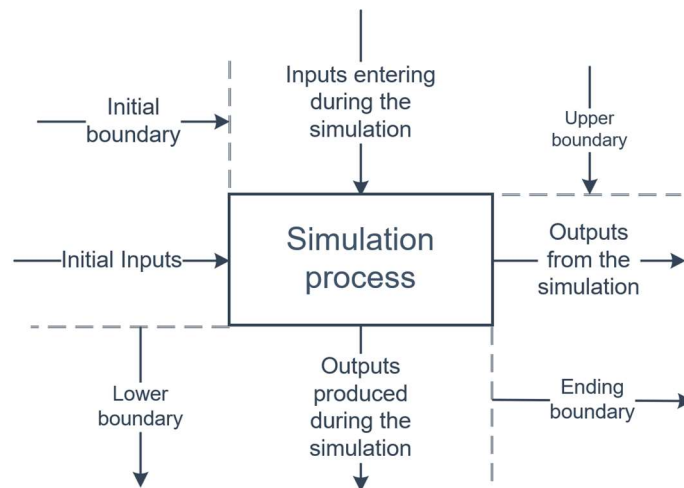


Figure 5.1.3.1 The four interaction points' boundaries, based on Mykoniatis and Angelopoulou (2020).

The *initial inputs* are the ones used to start the simulation and are at the *initial boundary* of the simulation model; the *outputs* of the simulation are the ones produced as final data or expected results and are at the end-ending boundary of the simulation system. *Works* produced during the simulation are those generated for different models or agents and are allocated in the *lower boundary of the model*. The inputs entering during the simulation come from other models or agents and are distributed in the upper limit of the simulation boundary. Communication between models of similar paradigms must be considered through the identification of variables properly “captured by” or “influenced by” (Mykoniatis and Angelopoulou, 2020).

5.1.4 Relationship definition.

Chahal (2010) identified four generic relationships between discrete events and system dynamics models usable to establish a the relationships definition, (1)To directly replacement of variables, (2) to summarize data or breaking down data in to

a component parts and (3) causal relationships Interactions that involve state changes injecting, adding or removing objects or entities, transfer entities, control flow statements, trigger events, and (4) state-chart control relationships are related to agent-based model interactions (Mykoniatis and Angelopoulou, 2020).

On a different note, Mykoniatis and Angelopoulou (2020) suggest the use of two categories to identify interaction points: Value assignment relationships and impact statements.

Value statements include mathematical formulations and the replacement of values between equivalent variables (Mykoniatis and Angelopoulou, 2020). On the contrary impact statement relationships cannot be expressed using value assignments, they are related to more abstract concepts. They may include one or more or a combination of value assignment relationships (Mykoniatis and Angelopoulou, 2020); in table 5.1.4.1, the categories' nomenclature and types and description of the interaction points are provided.

Table 5.1.4.1 Categories and types of relationships, based on Mykoniatis and Angelopoulou (2020)

Category	Types of relationship	Description
A. Value Assignment	Direct replacement of values of variables	Corresponds to interaction points that represent equivalent variables of information exchange in both models.
	Aggregation/dissagregation	Corresponds to interaction points that seize values of information exchange that need to be aggregated (accumulated) or disaggregated form the one model to equivalent values of the other model.
	Causal relationships	Correspond to interaction points described by explicitly mathematical relationships.
B. Impact Statement	Add/remove inject agents entities	Inject agents, in any point of the simulation process.
	Control flow	Correspond to "if" "for" and "while" statements and define the flow of a particular logic.
	Trigger event	Could be "timeout" "message", "condition" "rate" and arrival.
	State-chart control	Correspond to the state that may control the flow among two models, update variables from other models or trigger other relationship.

From the logical model construction, three main sections have been identified, the design task, the communications and trust and the workflow.

During the building intervention and evaluation (BIE) cycles of the Action Design Research method in Chapter 3, several iterations were carried out, implementing different aspects of the system process to understand the instrumentation of the modelling approach. Initially, the discrete events model of the product development process was built using different discrete events simulation approaches. However,

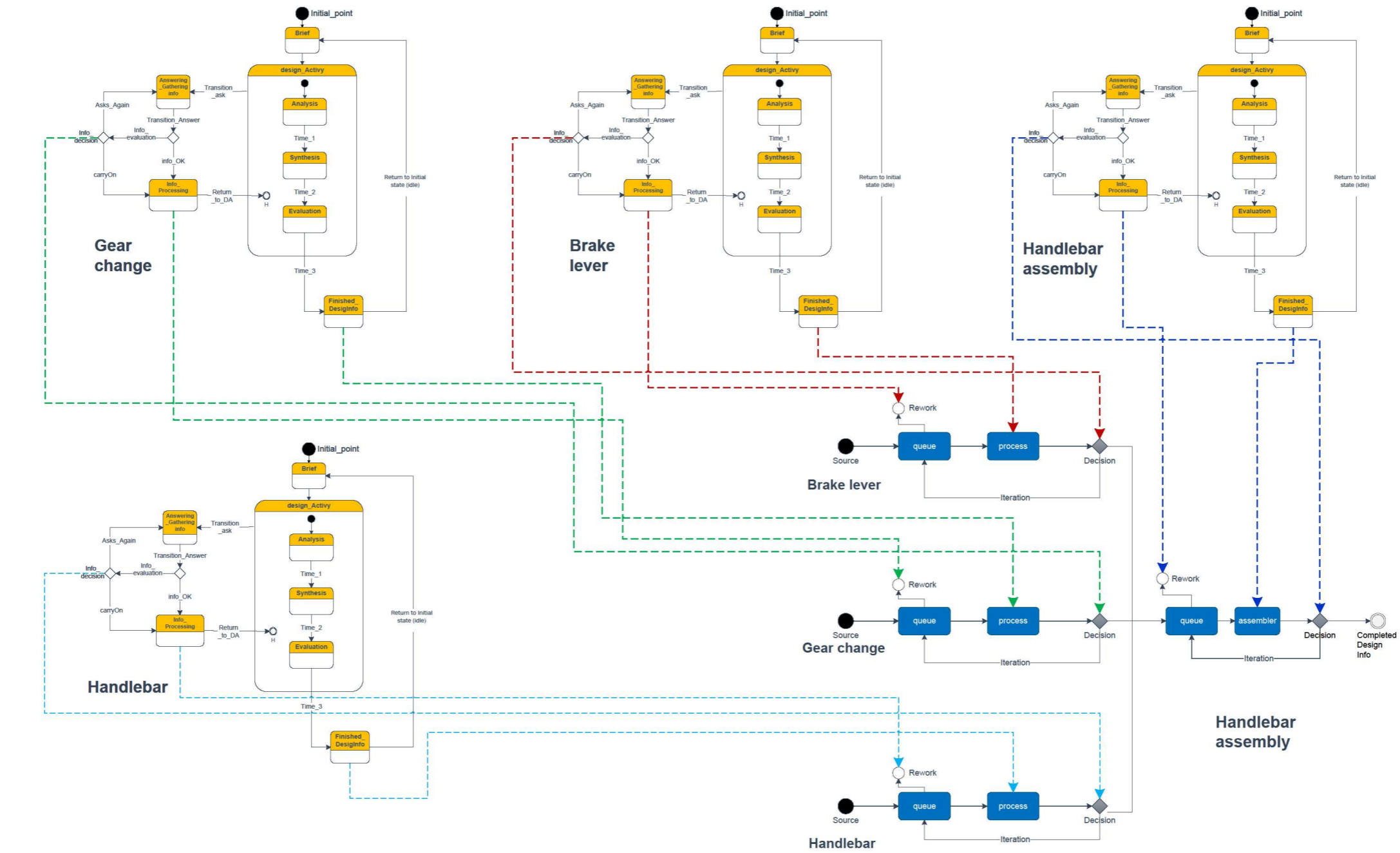
the Anylogic 8 software allowed the construction of a simulation model and offered possibly to include an agent-based and discrete events methods within the same simulation environment.

Those cycles consisted of elaborating different conceptual model in the system identification process and the system conceptualisation outcome in the form of diagrams, depicting the main system elements under study using the Unified modelling language (UML). The UML is a formal language used to specify and construct object-oriented systems that provide with a unified grammar to document and visualise the system elements and their relationships (Nikolic and Ghorbani, 2011). (Nikolic and Lukszo, 2013)

The integration of the agent-based with the discrete events model considers that each designer (agent) works in one of the identified parts of the bicycle handlebar assembly. The critical interaction points identified are between the end of the process of the agent-based model and the process (delay, service, or wait) block in the discrete events model. The interaction between them can be defined as a direct replacement of values because the process block is designed to seize the agent for a determined period; however, the seized time is determined for the time the agent-based takes to complete the activities until they arrive at the “finished_design” state. The following interacting points in the model are between the agent-based and the discrete events conceptual model's informing decisions in the agent-based to the decision blocks in the discrete events model. This relationship can be classified as a trigger event relationship since the change in the condition of the decision construct in the agent-based model produces an action in the decision block in the discrete events model. The next pair of interacting points is from the carry on construct in the agent-based model to the rework block in the discrete events model; this interaction can be identified as a trigger event relationship because it uses a condition to introduce a change in the decision block in the discrete events model.

Figure 5.1.4.1 shows the interaction points diagram between the discrete events and agent-based models. The second diagram in Figure 5.1.4.2 identifies the agent-based interaction points in state chart structures. There are three interaction points between agents; The transition_ask construct simulates the designer's request for information to the other designers through a message. In the Answering_gathering information state, the agent acknowledges the request and sends an answer to the requester, when the message is received in the state chart construct. The summary Transition_answer. the next state is the evaluation of the, where the agent decides to

ask for information. The summary of the identified interaction points in the UML hybrid model diagrams is depicted in table 5.1.4.2.



State chart diagram for agent based model

Figure 5.1.4.1 UML hybrid model diagram, showing interaction points between agent-based and discrete events model.

Table 5.1.4.2 Summary of interaction points in UML hybrid model diagrams

Agent/Part	Agent-based construct	Category	Type	events block	Description
(1)Brake lever, (2)Gear change, (3)Handlebar, (4)Handlebar assembly.	info_decision	Impact statement	Trigger event	Decision	Simulates the outcome of the information evaluation, and returns to the information request state when it is true. The cycles of information requests, answers and request information after evaluation simulate feedback loop iterations.
	Info processing (Carry-on)	Impact statement	Trigger event	Rework	Decision to carry on with available info. Simulates the chance that the designer decides to carry on with incomplete or defective information, resulting in general in rework.
	Finished design	Value assignment	Direct replacement	process	Simulates the end of the design activity, that delays the progression of the task in the discrete events model, the time taken to finish the process affects the time to complete the assembly.
Agent/Part	Agent-based construct	Category	Type	Agent-based construct	Description
(1)Brake lever, (2)Gear change, (3)Handlebar, (4)Handlebar assembly.	Transition ask	Impact statement	Trigger event	Anwering_gathering information	Simulates the need for information detected for the designer, sending request to other the designers
	Anwering_gathering information	Impact statement	Message	Transition Answer	Simulates when designer answers the request of information
	Transition Answer	Impact statement	Message	Anwering_gathering information	Simulates the reception of information from other designers and the transition to the information evaluation state

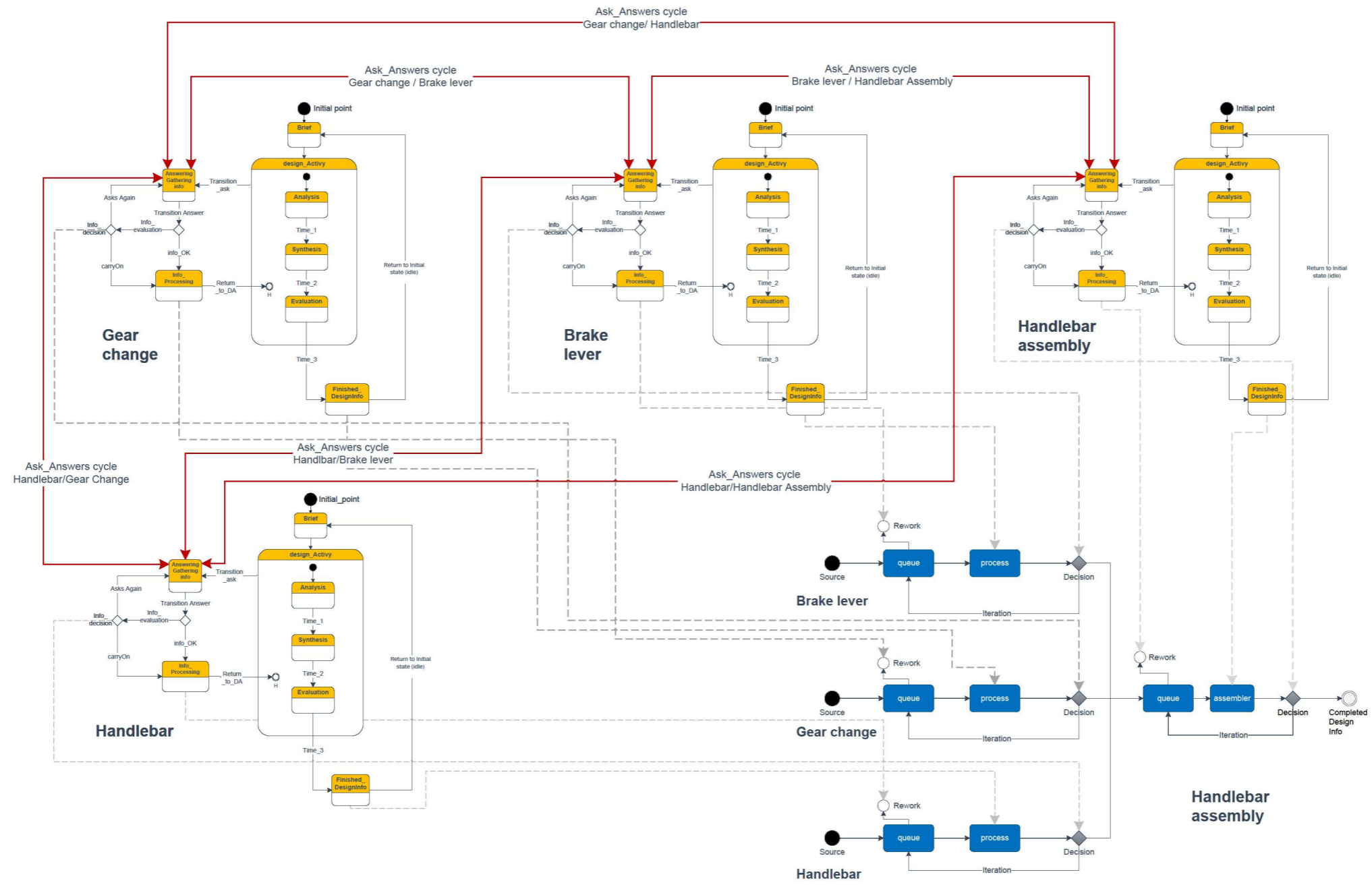


Figure 5.1.4.2 UML hybrid model diagram, showing interaction points between agent-based structures.

5. 2 Simulation.

In the modelling design stage, the identified concepts must be implemented in a computational language, retaining their original meaning (Nikolic and Ghorbani, 2011). In this stage, the research uses the meso-level approach, identifying the four components of the handlebar assembly and their relationships. The developed model combines a general-purpose design process framework and the product's architecture. Delineating specific feedback loops in the process that, in the context of the stage gates, reflect rework feedback loops or iteration feedback loops of individual designers and design teams.

Agents are the basic elements of an agent-based model. They can perform actions for themselves and other agents, receiving inputs from the environment and other agents and behaving flexibly and autonomously. Agents consist of states and rules (Nikolic and Lukszo, 2013). The agent states represent the specific parameters collection of all the relevant information about what the agent is now.

The rules describe how states are translated to action or new conditions. Rules should be understood as mechanical transformation functions rather than the colloquial use of the social notion of rules as regulations or agreements (Nikolic and Lukszo, 2013).

Actions are the actual actions agents carry out because of decision rules being applied to their states, and the behaviour is the total observable sum of the agent's activities. State changes are the agent's behaviour. It is an emergent attribute that results from the interaction of internal, local, and environmental states and decision rules.

The discrete events modelling visualises the systems as a series of sequential or parallel activities with delays and starting and endpoints. The suites for modelling in discrete events models include special blocks that allow merging the outcome from various individual processes into a new development; those blocks are usually named assemblers or assembly blocks. Some blocks can diverge entities from previous approaches; those blocks can accept, reject, or deviate entities with probabilistic or conditional criteria; in Anylogic eight, those blocks are called "decision" blocks.

In the conceptualisation of the handlebar design case study, the reception of a design requirement is identified as the process starting point. The product architecture defines the four actors performing design activities, and the progression from the individual parts to the assembly stage is conceptualised as the stage-gate process.












5.2.1 Discrete events modelling

The discrete events modelling visualises the systems as a series of sequential or parallel activities with delays and starting and endpoints. The suites for modelling in discrete events model include special blocks that allow merging the outcome from various individual processes into a new development; those blocks are usually named assemblers or assembly blocks. Some blocks can diverge entities from previous approaches; those blocks can accept, reject, or deviate entities with probabilistic or conditional criteria; in Anylogic eight, those blocks are called "decision" blocks.

In the conceptualisation of the handlebar design case study, the reception of a design requirement is identified as the process starting point. The product architecture defines the four actors performing design activities, and the progression from the individual parts to the assembly stage is conceptualised as the stage-gate process. The discrete events model implemented uses the input/source block to simulate the arrival of the design requirement. The design activities are modelled using operation blocks in Anylogic eight "service/delays" blocks. The "assembler" block merges the three individual parts in the handlebar assembly, and "decision" blocks are used to simulate iterations and reworks.

In Table 5.2.2.1, the initial parameters for the implementation of the simulation are shown; In the first column, the conceptual system elements are listed; in the second column, the correspondent Anylogic eight block used; in the third column, the simulation parameters used in the implementation of the discrete events model, and the fourth column the initial values of the simulation model.

Table 5.2.1.1 Initial parameters for the discrete events model.

Actors	Block	Description	Parameter	Initial value
Design request		Generates agents	Rate, per month	1~4 req. per month
Brake lever system design		Delays the agent	Delay time	Delay time, uniform distribution, uniform(4,7).
Gear change system design		Delays the agent	Delay time	Delay time, uniform distribution, uniform(4,7).
Handlebar system design		Delays the agent	Delay time	Delay time, uniform distribution, uniform(4,7).
Timeout for rework		Release the agent from the queue	Time	Uniform distribution; uniform(0,0.7)
Evaluation		Route agents to different flowchart branches	Percent	Percent(.4)
The handlebar assembly system		Allows a specified number of agents from several sources (5 or less) to be joined into a single agent.	Delay time	Delay time, uniform distribution, uniform(4,7).
Evaluation		Route agents to different flowchart branches	Percent	Percent (.1) (3x)
Resource pool		Define a set of resource units, used to represent staff, or equipment	Capacity	1~4 units
Queue		Works as a buffer for agents to wait to be accepted in the next block	Maximum capacity	N/A
Finished design		Disposes agents	N/A	End of the process
Simulation parameters				
Model Time		Model stops at specific time		156 weeks
Randomness		Unique simulation runs		Random seed

The simulation model considers how to design iteration rework and how interactions within and across design teams influence the progression and efficiency of design tasks in the new product development systems.

In the simulation model (Figure 5.2.1.1), design requirements are fed into the model and transformed into finished designs. The number of feedback loops influences the number of designs produced. During the simulation, iterations are randomly selected feedback loops that occur within the design process of a given part. In contrast, rework loops span the design process of multiple components and are modelled as service delays to queues. Together, these

feedback loops influence the time taken to produce a design, and so the number of designs created in a simulation experiment with a fixed runtime.

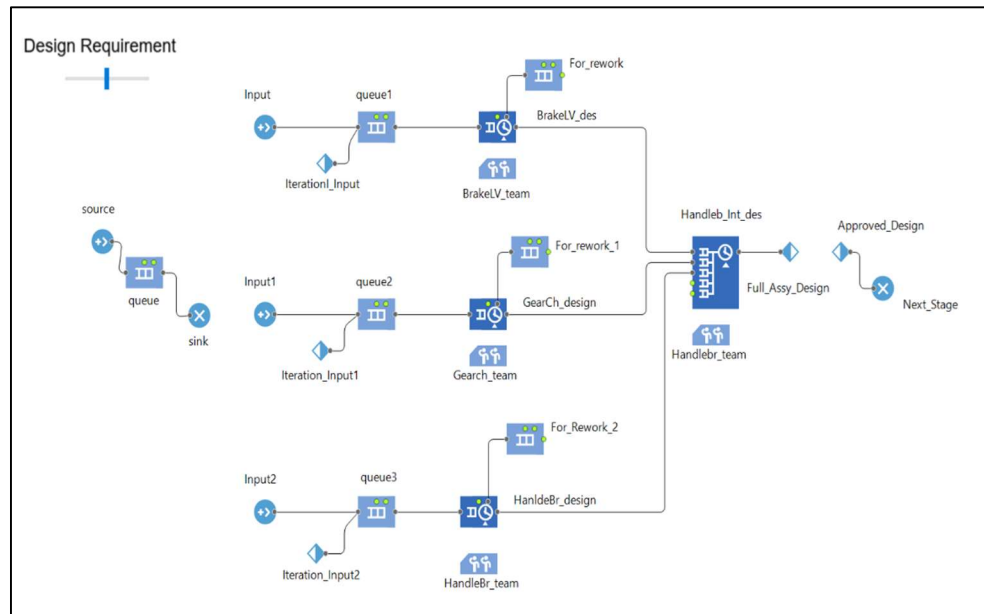


Figure 5.2.1.1 Discrete events simulation model, implemented in Anylogic 8

The initial input parameter of the simulation model is the number of design requirements. The assumptions are that the time taken for a design team to complete the design task is between 4 and 7 weeks, and the amount of iteration allowed is about 30 per cent. The simulation output is the number of designs produced in the runtime experiment, which affected the number of iterations and volume of rework given a fixed capacity.

5.2.2 Agent-based

Agent-based simulation models a system as a collection of autonomous individuals called agents that individually evaluate and make decisions based on a set of rules and execute various behaviours appropriate for the system they represent (Bonabeau, 2002). Agent-based simulations contribute to the construction of models without knowledge about the global interdependences or the global sequence of operations. Those models can be constructed by perceiving how individual participants of the process behave, obtaining a global behaviour (Borshchev & Filippov, 2004). The intervention artefact of this iteration was developed using eight. This package uses state charts to model system behaviours that cannot be defined using events or dynamic events. The Anylogic state charts have states and transitions triggered by user-defined conditions (timeouts or rates,

messages received by the state chart, and Boolean conditions) (Anylogic, 2022). Figure 5.2.2.1 illustrates the state charts construct elements from Anylogic eight.

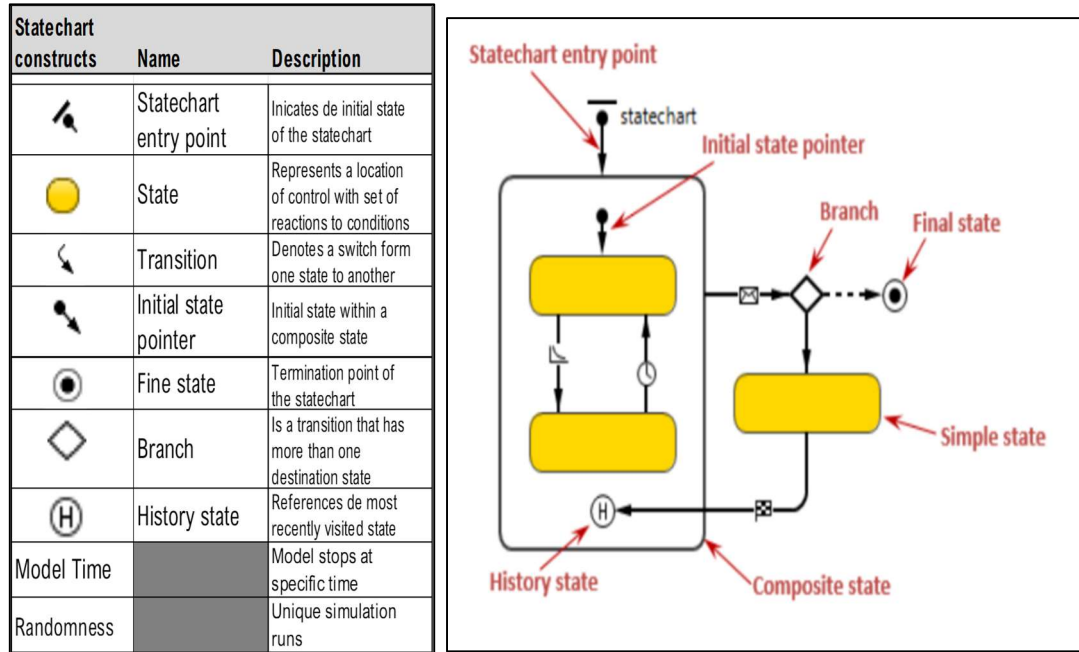


Figure 5.2.2.1 State chart constructs from Anylogic 8 (2022).

The system identification and conceptualisation outcome of the engineering design, the information processing and communication is a conceptual model represented using a UML machine states diagram shown in Figure 5.2.2.2

At the centre of the diagram, the designer moves to the state of processing information or communicating alternatively during the design activity. While the designer is in the design activity state it goes through the stages of analysis, synthesis, and evaluation. Within each step, designers may require information from the other design teams working in different parts of the system.

To acquire the information, designers (agents) transit from the design task state to the communication state and from there to the information processing state to return to the same point they left in the design task. During the information processing, the agent evaluates information and decides whether to ask for more details (iterates) or to complete the job with incomplete or missing information.

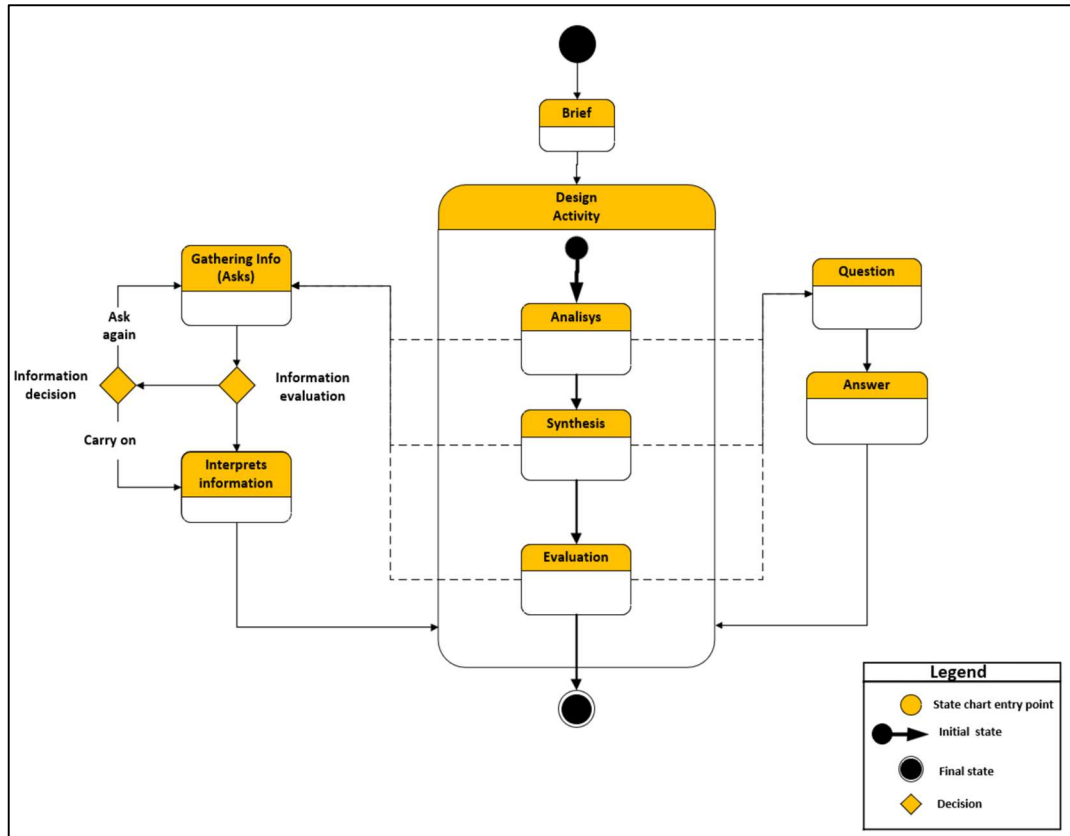







Figure 5.2.2.2 UML State chart diagrams for the agent-based conceptual model.

The state chart constructs and parameters used in the simulation model of the designer agents are depicted in Table 5.2.2.1 The first column identifies the correspondent Anylogic eight state chart constructs. In the first column, the description starts with the states, followed by the transitions, decision blocks, and history blocks and finalises with the identification of the variables. The second column shows identification names resulting from the system conceptualisation process but adds other constructs necessary for the configuration of the simulation. The third column contains a brief description of the construct element function, followed by the column of the definition of the parameters; here, parameters are defined as time scales, week, month, year, or Boolean values like true or false.

The triggers used in transitions define the model behaviour, changing the agent state or creating conditions for a state change. Not all the constructs have implicit triggers or parameters; some constructs are used to support the flow in the simulation diagram. Finally, the last column contains the parameters used in the simulation. Those parameters are assumptions, i.e.,

the time to complete that design task is estimated in a period of 4 to 7 weeks; this time is divided into the three design stages identified, giving 20 per cent to the analysis activity, 50 per cent to the synthesis stage and 30 per cent to the design evaluation stage.

Table 5.2.2.1 Initial parameters for agent-based simulation.

State	Given name	Description	Parameter definition	Trigger	Action parameters
	Brief	Initial designer state			
	Design_activity (composite state)	Includes design stages			
	Analysys	Design stage			timeOfEnTotal =time()
	Synthesis	Design stage			
	Evaluation	Design stage			
	finished_Design	End of the process			Finished_desgin++
	Answering_Gathering_Information	Gets information/Gets a question			
	info_processing	Works with the information			
	rwork	Rework process			
Transition					
	Design_req	Arrival of design requirements	Month	Rate	1,2,3 or 4; Requirement++
	Time_1	Time to complete the phase	Week	Timeout	uniform(4,7)*.2
	Time_2	Time to complete the phase	Week	Timeout	uniform(4,7)*.6
	Time_3	Time to complete the phase	Week	Timeout	uniform(4,7)*.3
	transition_Ask	Designer realises need for information		Condition	randomtrue(.15)
	transition_Answer	Designer receive an answer	Week	Timeout	
	info_OK	Information is adequate	TRUE	Condition	
	info_evaluation	Decides on information		Condition	randomtrue (.3)
	ask_Again	Request for clarification or asks again	TRUE	Condition	Iteration++
	carryOn	Decides to carry on with incomplete or without information		Condition	randomtrue (.3); Rework++
	return_	Returns the model to the initial state	Day	Timeout	1
Decision					
	info_rev	Review information		FALSE	
	info_decision	Decides to carry on with incomplete or without information		FALSE	
	History State	Returns to the previous state where the process left.			
Variables					
	timeinEsTotal				
	timeOfEnTotal				
	Requirement				
	Iteration				
	rwork				
	Finished_Design				
Simulation parameters					
Model Time			Model stops at specific time	156 Week	
Randmonees			Unique simulation run	Random seed	

The need for information transition has an essential role in the simulation model. Together with the time between stages, determine the simulated system's efficiency. Figure 5.2.2.3 presents the state chart diagram of the simulation structure developed in Anylogic eight. The brief state is used as an Idle state, so the model initialises when the transition Design_req triggers the state change to the Design_Task composite state; there, the other states and transitions start interacting. In the analysis stage, the state change is triggered by the Time_1 transition, repeating a similar pattern going through

the Synthesis and Evaluation states until the system reaches the finished_Design state and returns to the idle state in the Brief state. However, when the transition_Ask triggers the state change. The simulation goes through cycles of asking for information, evaluating information, asking again, simulating iteration feedback cycles or deciding to carry on without incomplete information, and returning through the History construct to the design_Task until the process is complete. The difference between the conceptual model and the simulation state chart is due to the simulation constructs' configuration requirements.

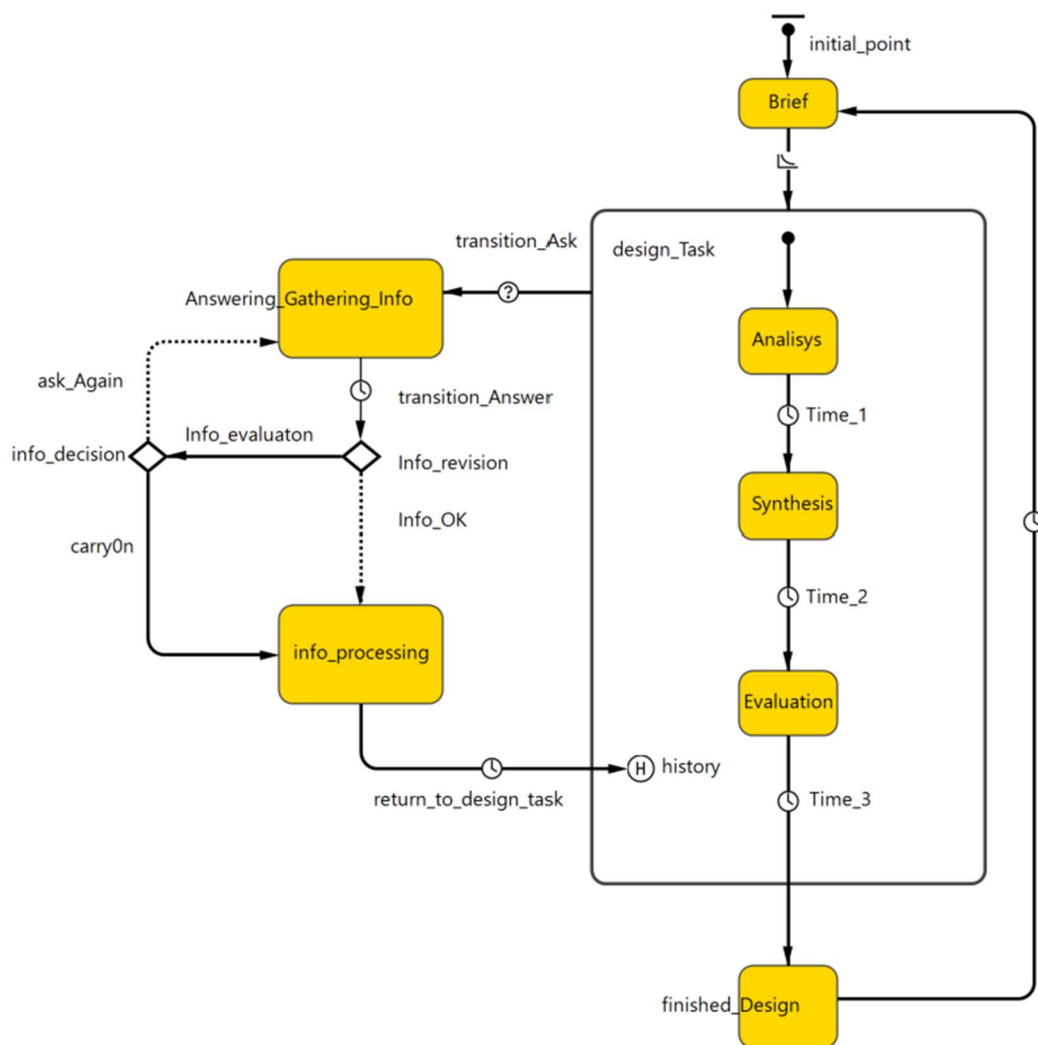


Figure 5.2.2.3 Agent-based simulation diagram implemented in Anylogic 8

Similar to the discrete events model, the initial input parameter of the simulation is the number of design requirements; in the case of the agent-

based model, requirements can be set up to one to four per month in each experiment. The assumptions for the simulation parameters are the time taken for a design team to complete the design task is between 4 and 7 weeks. The amount of iteration allowed in this case ranges from 15 per cent to .30. The simulation output is the number of designs produced in the runtime experiment, which is affected by the number of iterations and volume of rework given a fixed capacity.

5.2.3 Hybrid Simulation, Discrete events+Agent-based

Discrete events modelling is often considered as a list of events to be processed, or a flowchart with entities and mobile resources flowing through the processes (Goh and Ali, 2016) in this approach entities and resources are not able to interact with each other and they do not display adaptive behaviours (Scheidegger et al., 2018). On the other hand, agent-based modelling is a bottom up approach focused on designing individual agents able to make decisions and perform actions, and their emergent behaviour arises from those interactions (Borshchev, 2013; Dubiel and Tsimhoni, 2005).

The combination of two or more simulation approaches leads to what is called hybrid or multiparadigm simulation modelling. Scheidegger et al. (2018) argues that the integration of two or more simulation methods, is useful to develop simpler and more efficient models. Because through those approaches is possible to gain a better understanding of complex systems with different dimensions and perspectives. (Scheidegger et al., 2018).

The Multi-Paradigm Modelling methods enable to model each aspect of a system explicitly at the most appropriate abstraction level (Challenger et al., 2020), allowing the generation of an interoperable simulation able to capture interactions among elements (Mykoniatis and Angelopoulou, 2020) identified in the Section 3.2.2 of the methodology chapter.

The integration of two simulation methods suggested in the conceptual framework, captures the social and behavioural patterns of the actors, the effects on the progress of design work and the process. Including, the logical and chronological structure of the project and the product architecture.

The integration of the agent-based with the discrete events model considers that each designer (agent) works in one of the identified parts of the bicycle handlebar assembly, so there is one agent for the brake lever, one agent for the gear change, one for the handlebar and on for the handlebar assembly.

Each agent has three main interaction points with the correspondent branch of the discrete events model.

First, between the end of the agent-based model and the process (wait) block in the discrete events model the time taken for the designer agents to process the design requirement determines the time to pass to the next block. The interaction between them can be defined as a direct replacement of values because the process block is designed to seize the agent for a determined period; however, the seized time will be determined for time to complete the design task of designer agents.

The second relevant interaction point is the between the “carry on” which has direct influence in the “rework” block in the discrete events model. The carry-on of each individual part evaluated in handlebar assembly agent-based model triggers the “timeout” function in the wait (process) block in the discrete events model, simulating the rework that will affect a previous stage in each branch of the discrete events model. In the hybrid model this relation is identified as impact statement, triggering the “timeout” function in the wait (process) block.

In the third interaction point, individual part agents (designers) influence their respective “decision” block for simulate the iterations in the discrete events model. These interacting points are impact instatements triggering the actions.

The agent-based model has three interactions points that simulate the information and communication patterns between each agent (designer)

First, ask question, agent (designer), asks for information to other agents, second agents answer the question, and third the agent receive answer and move the next state, those three interaction points are in the same category impact statement, type message. The table 5.2.4.1 summarizes the main interaction points identified in the hybrid simulation implementation.

Table 5.2.3.1 Summary of the interaction points identified.

Agent/Part	Agent based construct	Initial boundary	Upper boundary	Lower boundary	Ending boundary	Category	Type	Discrete-events block	Initial boundary	Upper boundary	Lower boundary	Ending boundary	Description
(1) Brake lever, (2) Gear change, (3) Handlebar, (4) Handlebar assembly.	Finished design				X	Value assignment	Direct replacement	Wait, (release wait)		X			Simulates the end of the design activity, that delays the progression of the task in the discrete events model, the time taken to finish the process affects the time to complete the assembly.
	Carry-on			X		Impact statement	Trigger event	"Timeout" in Wait (Rework)		X			The decision to carry on with available info. This simulates the chance that the designer decides to carry on with incomplete or defective information, in general resulting in rework.
	Information evaluation			X		Impact statement	Trigger event	Decision block		X			Triggers the timeout function. Simulates the detection of rework that will affect a previous stage.
(1) Brake lever, (2) Gear change, (3) Handlebar, (4) Handlebar assembly.	Ask Question			X		Impact statement	Message	Answer Question		X			Simulates the need for information detected for the designer, sending request to other the designers
	Answer Question			X		Impact statement	Message	Have answer		X			Simulates when designer answers the request of information
	Have answer			X		Impact statement	Message	Evaluates information		X			Simulates the reception of information from other designers and the transition to the information evaluation state

5.3 Validation

Nikolic and Ghorbani, (2011) assert that several experimental methods, are used to explore the parameters space of the model to search for the optimal configuration that has been developed. Parameter sweeps consist in systematically adjusting model parameters to investigate as many combinations possible(Nikolic and Ghorbani, 2011). Each test parameter has a start, end, and increment value that sets the parameter's range of potential values. They also suggest the parameters variation experiment, similar to the parameters sweeps method mentioned above, includes the parameters' start, end, and increment values and allows the configuration of simulation models comprising several single models runs, varying one or more parameters (Nikolic and Ghorbani, 2011).

The evaluation of the BIE assemblies artefact as a simulation model uses an experiment performed in three sets of 20 simulation runs, The required parameter is changed in each experiment, which helps to observe how changes in the workload of the designers' teams impact the number of iterations and rework implemented. The plots created from the simulations are shown in Figures 5.3.1.1.

5.3.1 Validation of the Discrete Events Model

The first experiment runs a single requirement per month, with the simulation stopping at 156 weeks, and using the option of "unique simulation runs" shows that from a total of 36 requirements, 20 requirements were completed in time and without iterations or rework, 15 requirements were iterated and from those three were reworked (shown in plot "a"). In the second plot, the parameter for design requirements was set to two per month for 156 weeks The system received an average of 76 requirements, from which 43 were completed in time, 31 were iterated, and seven were reworked (see plot "b"). For the final experiment (plot "c"), the input was three monthly requirements for 156 weeks of runtime. The experiment shows a noticeable change in the rework. Of the 112 requirements accepted, 53 were completed in time, 41 required iterations, and 52 required reworks.

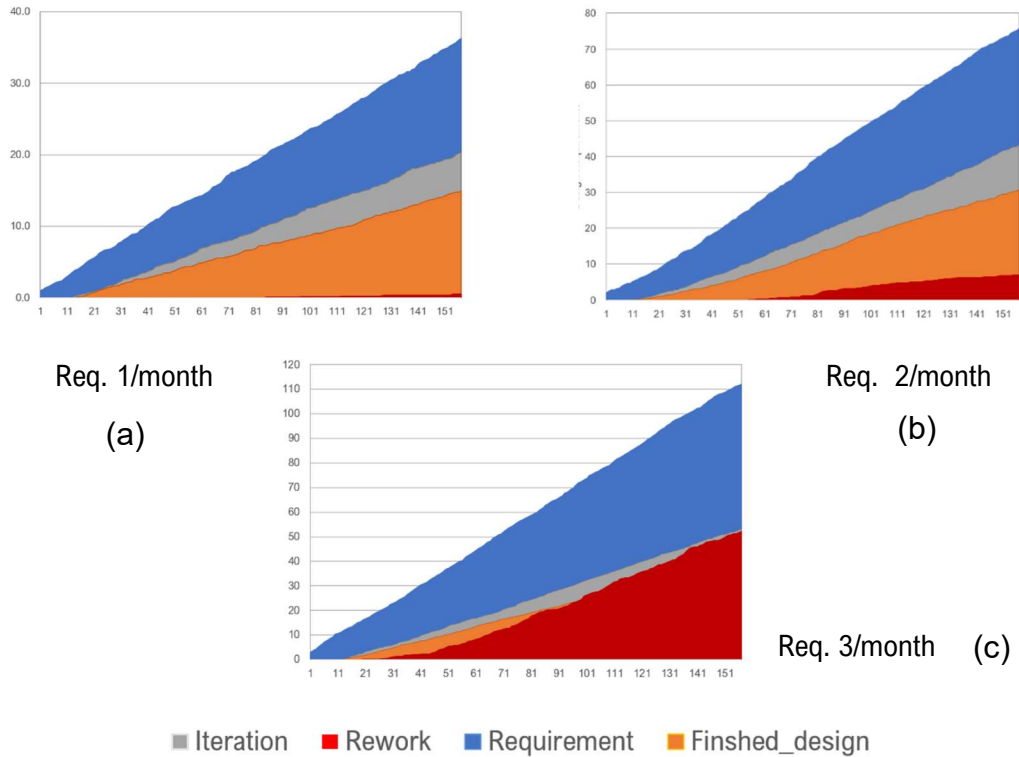


Figure 5.3.1.1 Time plots showing results of the discrete events simulations.

The model development allowed the identification of the task's temporal aspects and the system's conceptualisation, where iterations and rework are characterised as random events occurring during the execution of the design process. In the configuration and characteristics of this artifact, design is highly influenced by the number of design requests and the number of information requests received to the point that the number of requests produces the same number of reworks.

5.3.2 Validation of the Agent Based Model

Similar to the discrete events model, the initial input parameter of the simulation is the number of design requirements; in the case of the agent-based model, requirements can be set between one to four per month in each experiment. The assumptions for the simulation parameters are that the time taken for a design team to complete the design task is between 4 and 7 weeks. The amount of iteration allowed in this case ranges from 15 per cent to 30. The simulation output is the number of designs produced in the runtime experiment, which is affected by the number of iterations and volume of rework given a fixed capacity.

The evaluations of the simulation experiment presented in Figure 5.3.2.1 are performed in four sets of 10 simulation runs, changing the transition_Ask parameter in each experiment, which helps to observe changes in the designers' workload, impacting the number of finished designs, iterations and rework. The plots reporting the result of the four variables investigated; (accepted requirements, iterations, reworked, and the finished design) is shown in the figures below. The first experiment (shown in a) runs the transition_Ask at randomTrue (0.15), with the simulation stopping at 156 weeks and using the “unique simulation runs” option”, which shows that the system can produce an average of 13.3 requirements.

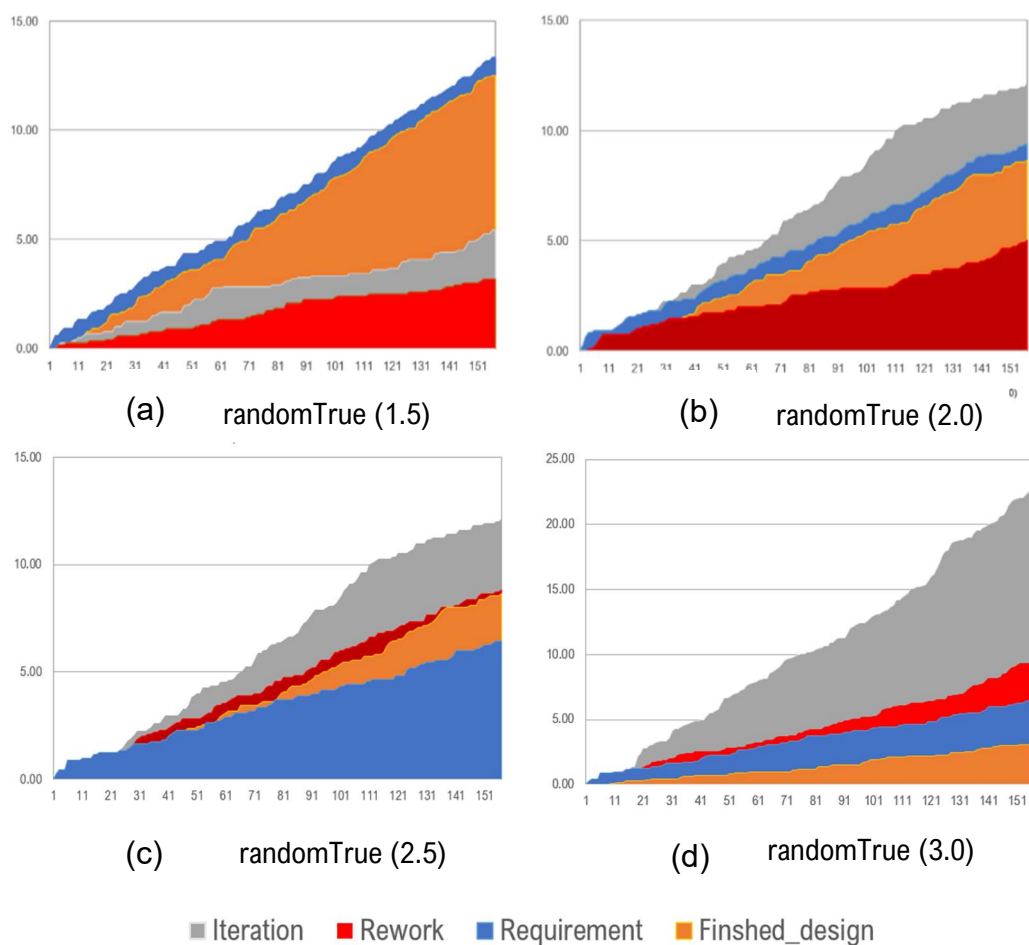


Figure 5.3.2.1 Time plots showing the results for agent-based experiments.

From there, an average of 5.4 were iterated, and from those, 3.17 were reworked. The outcome was an average of 12.50 finished designs. In the plot b, the system processes an average of 9.36 requirements. From which 8.64 were completed in time, 12.18 were iterated, and five were reworked. In

the following experiment shown in plot c, the transition_Ask was set to randomTrue 0.25, producing the following results. The requirements average was 6.45, and the average of iterations was 15.73, having an average of 8.82 reworks and a total of 5.50 finished designs at the end of the 156 weeks of the simulation run.

The evaluation experiments used the same input of two requirements per month in 156 weeks of runtime. However, the transition_Ask trigger parameter was set to randomTrue 0.3 presented in plot d. There is a noticeable change in the average number of iterations and the finished designs. From an average of 4.00 requirements accepted, an average of 3.09 were completed, 23.09 iterated and 9.45 reworked. For all experiments, the info_evaluaton parameter was set to randomTrue (0.3).

The agent-based simulation model considers design iteration rework, and the interactions influence the progression and efficiency of the design process in the simulation model. The number of designs produced is affected by the number of feedback loops and the time set in the transitions between the design stages of analysis, synthesis, and evaluation. Iterations are random feedback loops within the information evaluation process, whereas rework loops result from a lack of communication between the agents.








5.3.3 Validation of the hybrid simulation.

The simulations experiments used for validation of the framework use a distinct set of parameters, first the model time was set to 312 weeks (6 years) doubling the time of the previous experiments, in consequence the number of runs to 20. There is an increment in the range of Ask, Evaluation and Carry on parameters looking to reflect a mayor impact in the system.

Table 5.3.3.1 summarizes the parameters used in the validation experiments.

Table 5.3.3.1 Validation experiment parameters.

Validation hybrid simulation experiment parameters										
Exp No.	Requirement	Design Task Time	Model time	Randomness	Ask	Evaluation	Carry on	Number of runs	Scatter x,y plot	Staked plot
1	3 parts per year (one part each 18 weeks)	Uniform (3,7) Weeks	312 Weeks	Random seed	0.25	0.35	0.45	20	a,1	a,2
					0.35	0.45	0.25		a,3	a,4
					0.45	0.35	0.25		a,5	a,6
2	4 parts per year (one each 12 weeks)				0.25	0.35	0.45		b,1	b,2
					0.35	0.45	0.25		b,3	b,4
					0.45	0.35	0.25		b,5	b,6
3	6 parts per year (one each 9 weeks)				0.25	0.35	0.45		c,1	c,2
					0.35	0.45	0.25		c,3	c,4
					0.45	0.35	0.25		c,5	c,6

Legend	
	Evaluation
	Iteration
	Rework
	Brake lever
	Gear Change
	Handlebar
	Handlebar assembly

The design requirements for the development system experiments will vary the quantity of experiments between three design requirements per year and six requirements per year. The parameters variation experiment changes the probabilistic chance to require information (ask), the probabilistic chance to evaluate to ask again (iteration) and the probabilistic chance to carry on without the information (rework). The data of the experiments are linked, the simulation run that produced scatter plot data, also produced the data for the staked plot. For instance, the data presented in a,1 is linked to the data in a,2. The plots for the simulation runs are presented in following graphs numbers from 5.3.3.1 to 53.3.18,

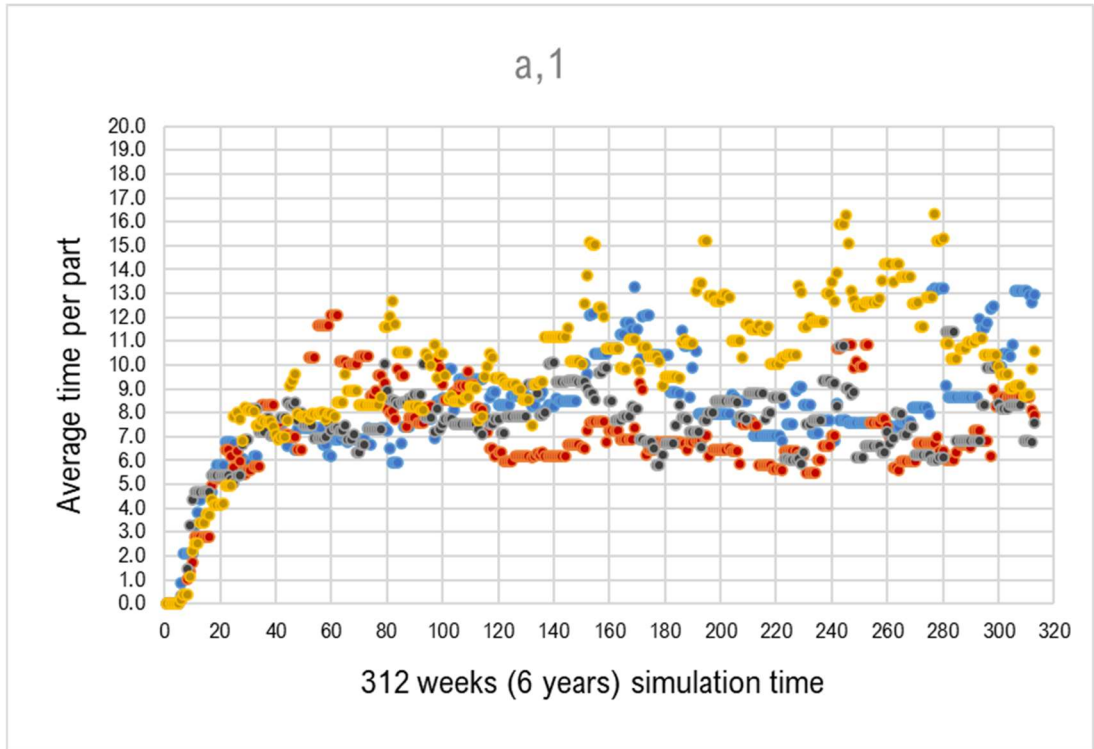


Figure 5.3.3.1 a,1 average time per part designer

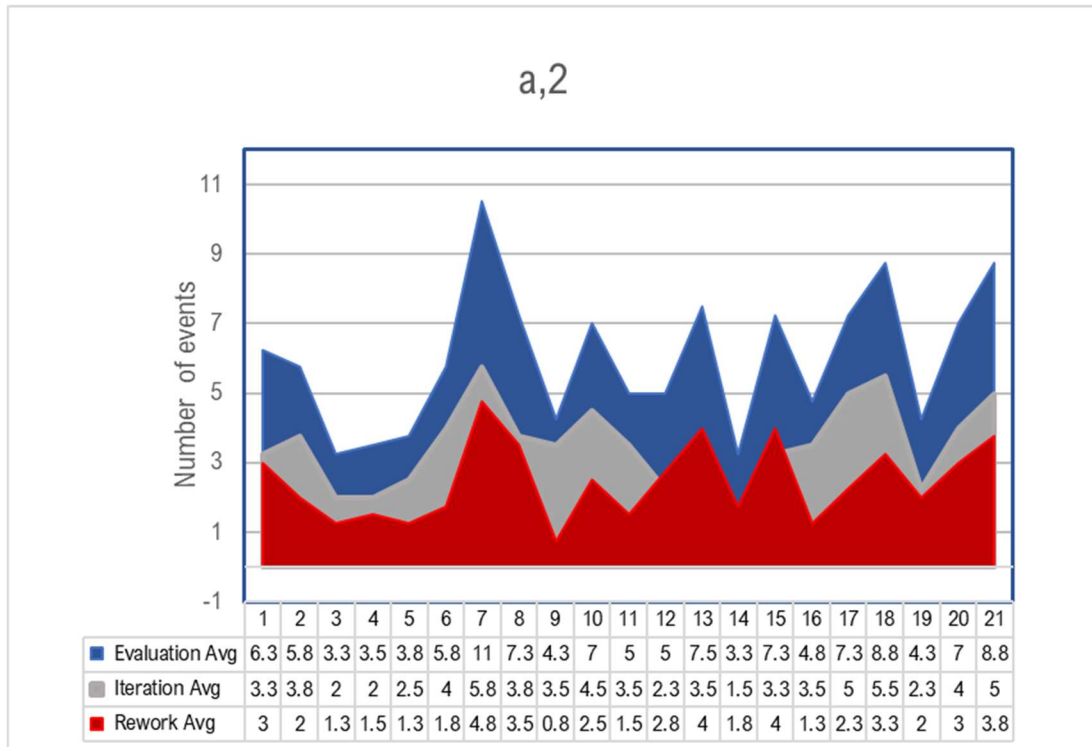


Figure 5.3.3.2 a,2 average number of events per run

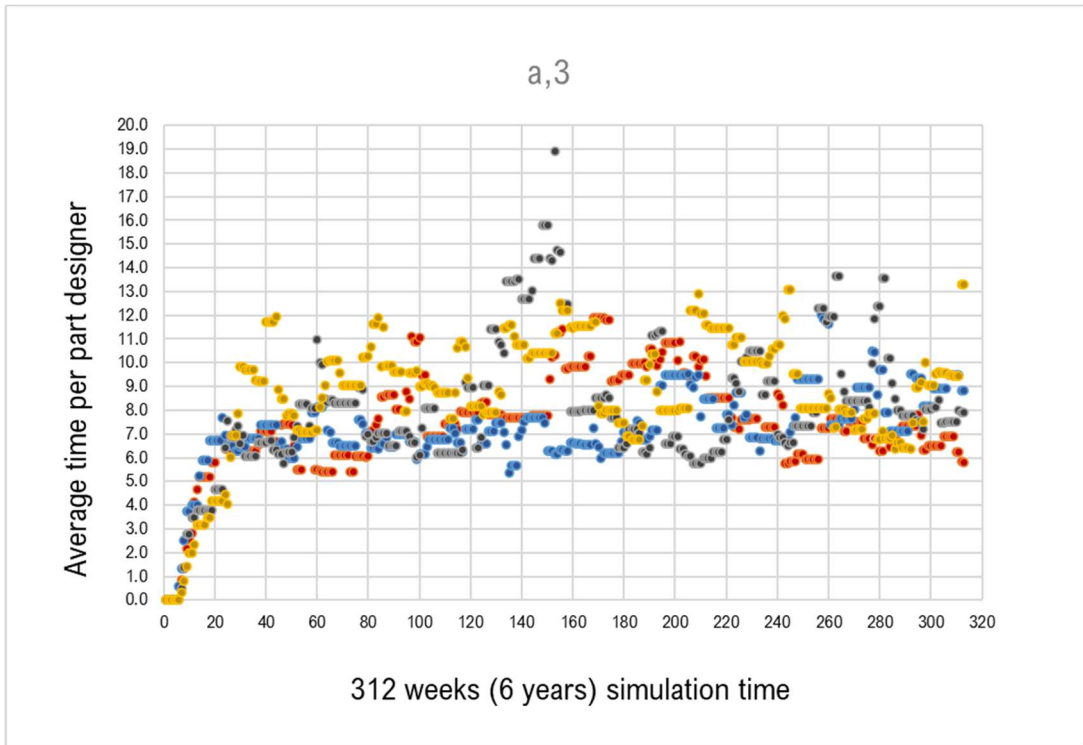


Figure 5.3.3.3 a,3 average time per part designer

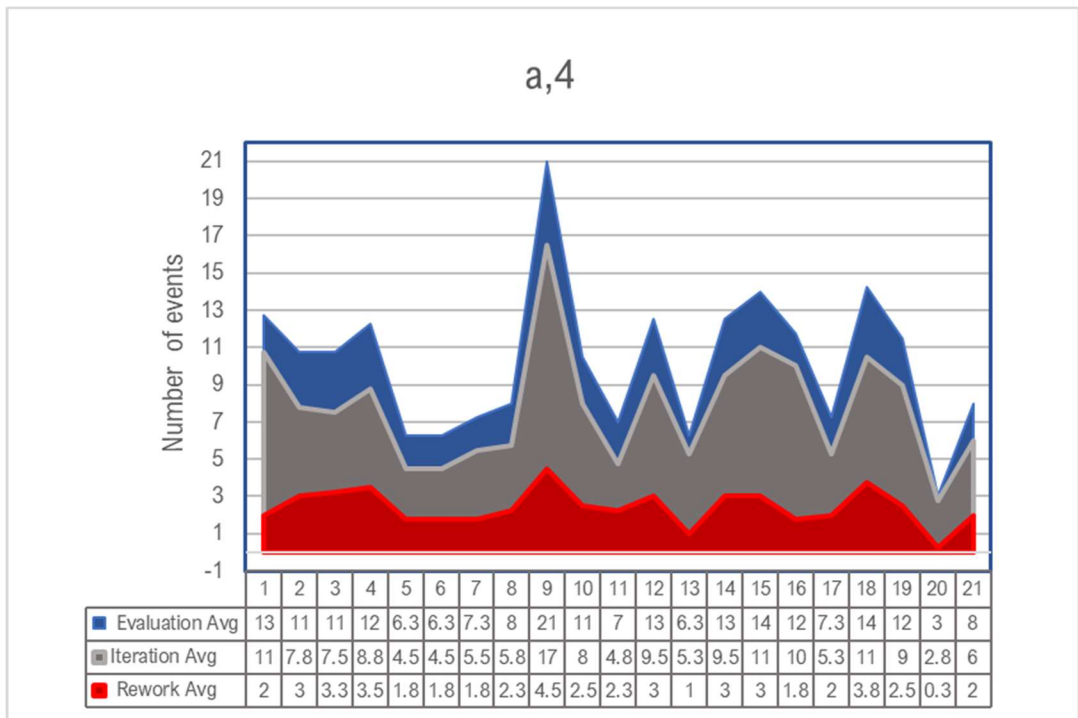


Figure 5.3.3.4 a,4 average number of events per run

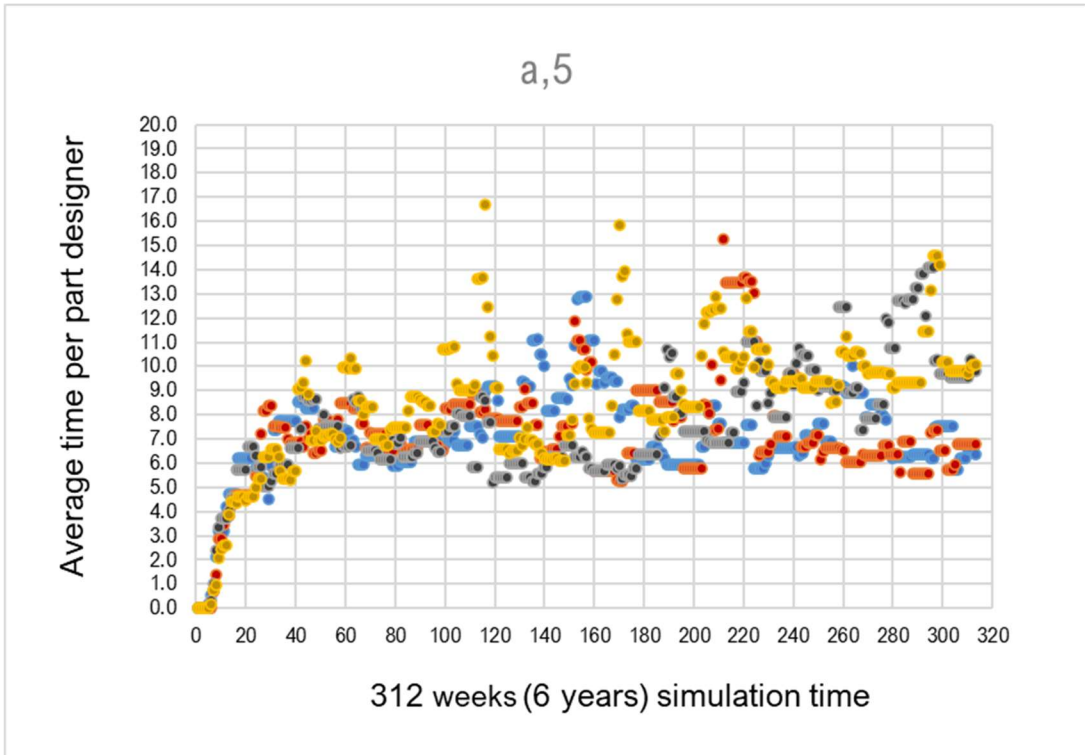


Figure 5.3.3.5 a,5 average time per part designer

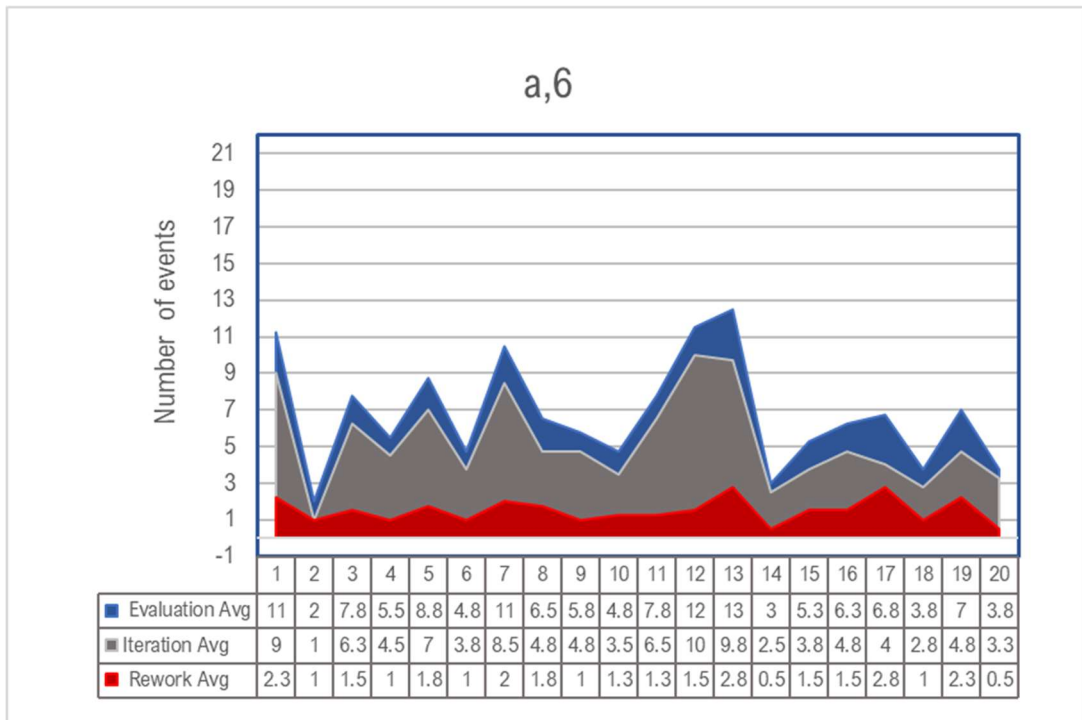


Figure 5.3.3.6 a,6 average number of events per run

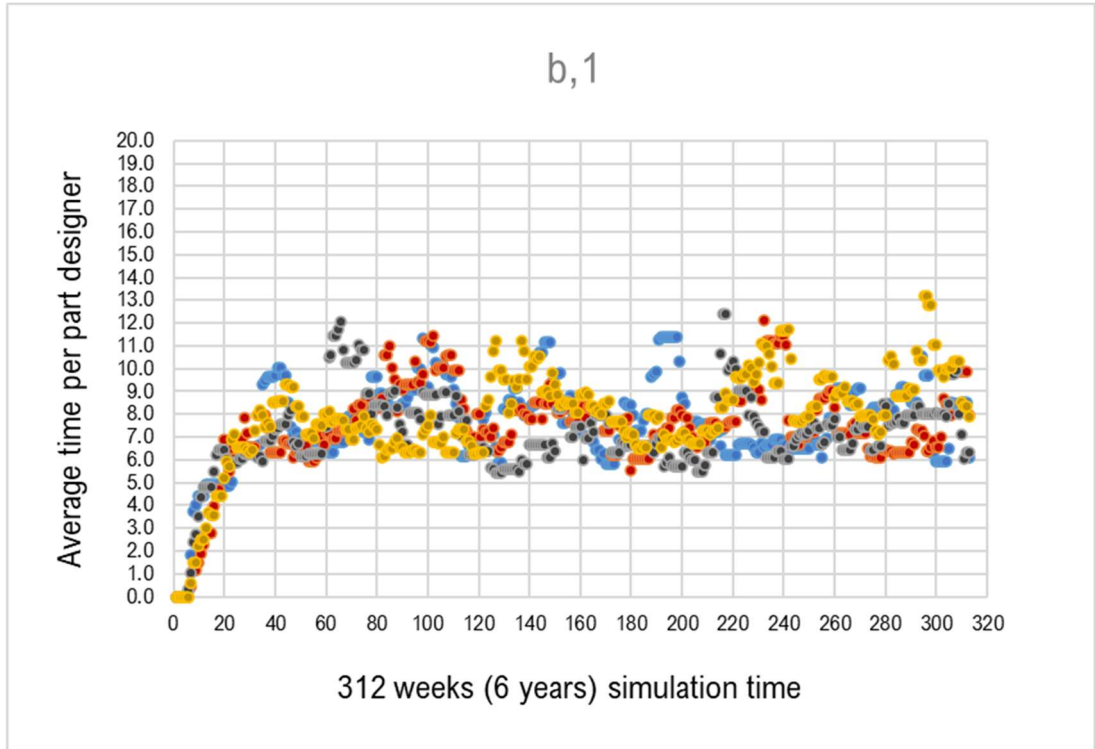


Figure 3.5.5.7 b,1 average time per part designer

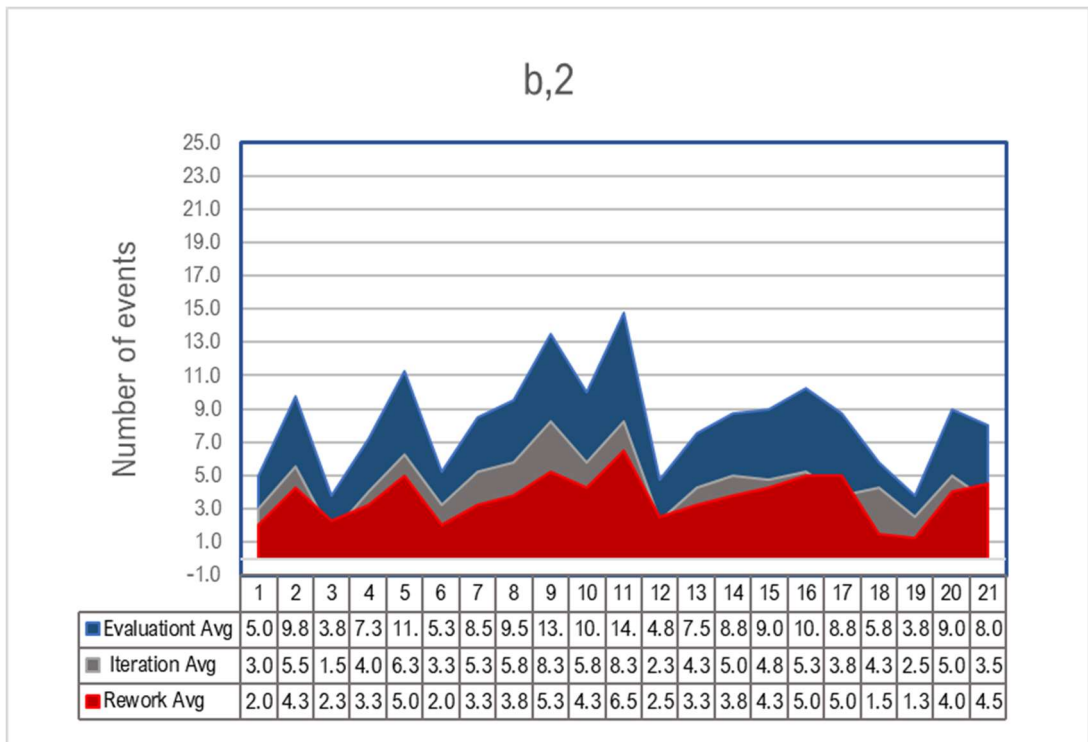


Figure 3.5.5.8 b,2 average number of events per run.

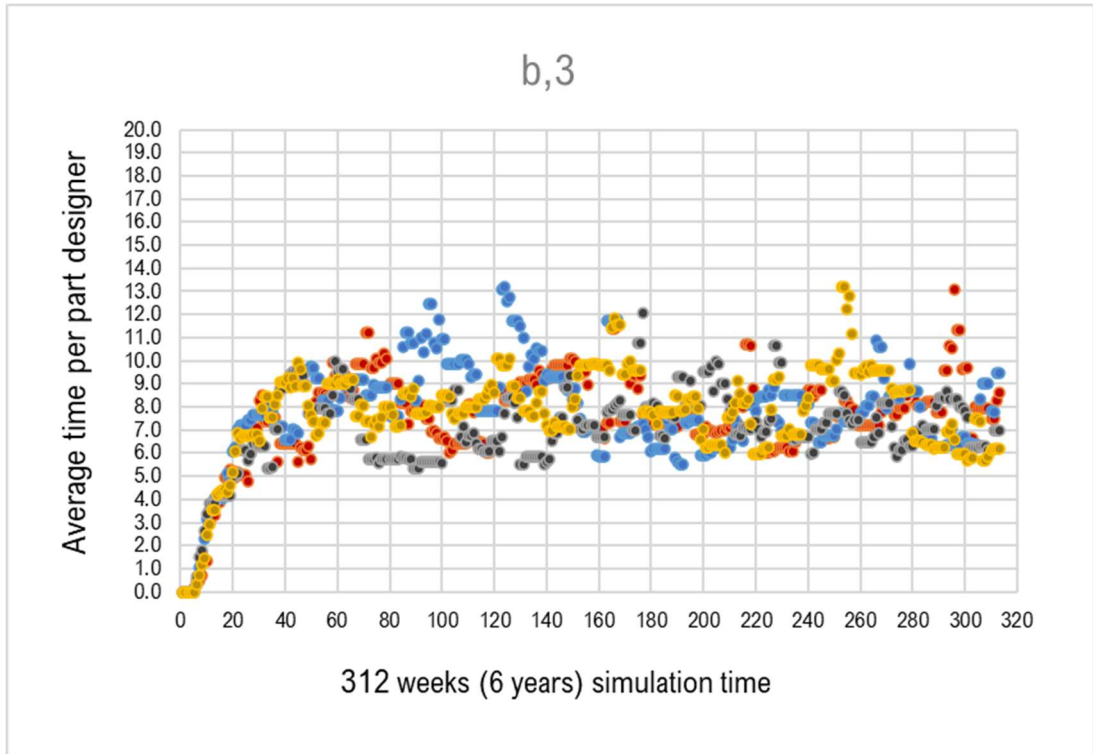


Figure 5.3.3.9 b,3 average time per part designer

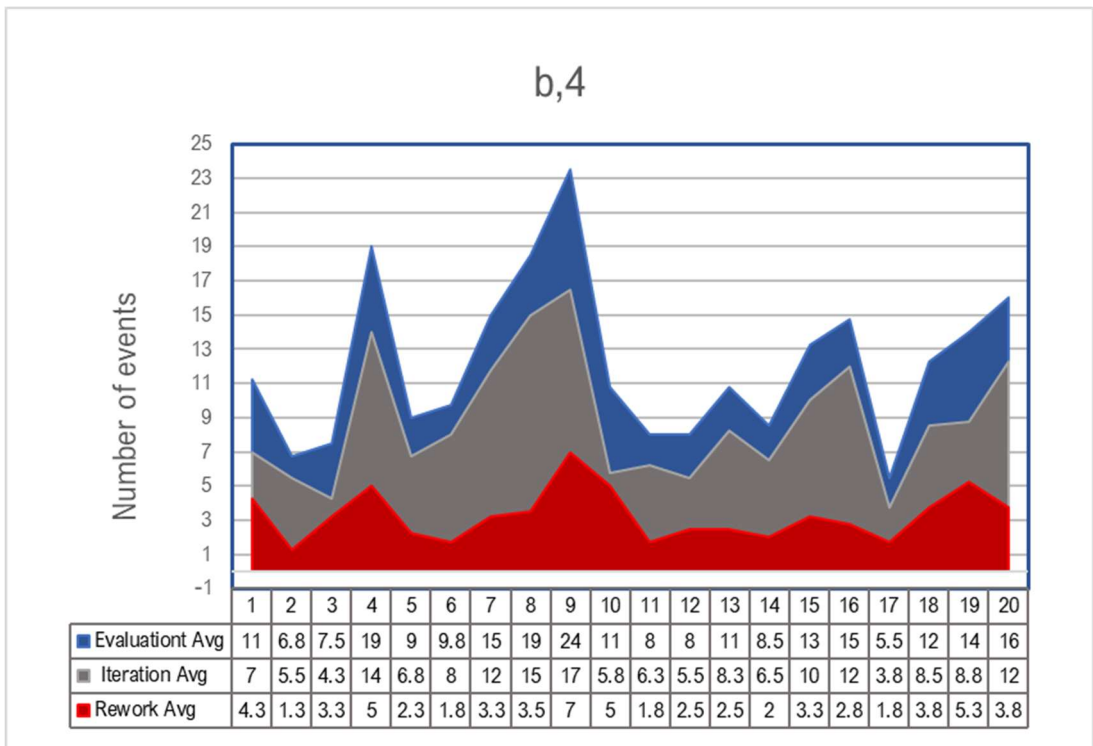


Figure 5.3.3.10 b,4 average number of events per run

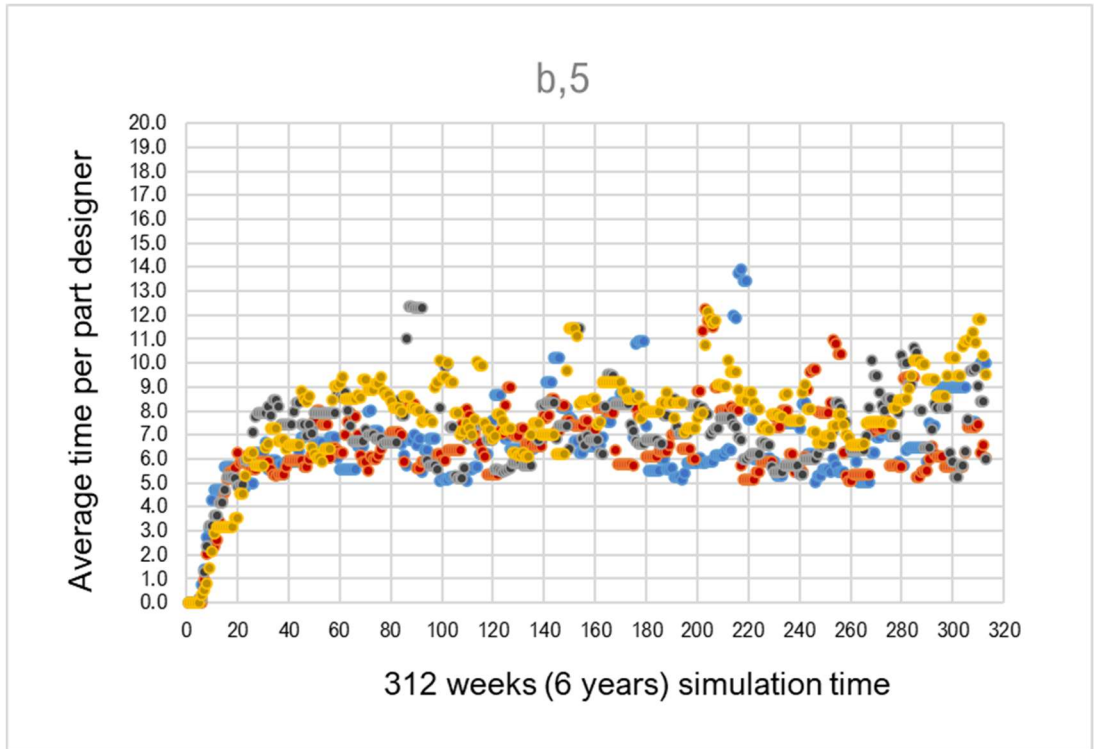


Figure 5.3.3.11 b,5 average time per part designer

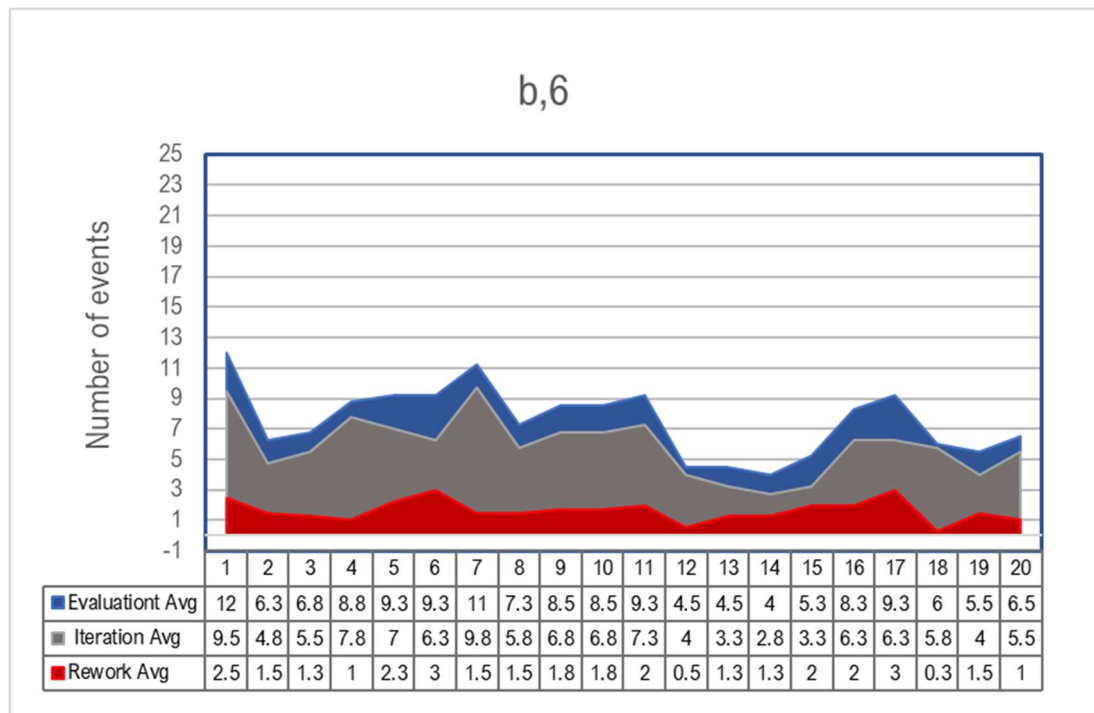


Figure 5.3.3.12 b,6 average number of events per run

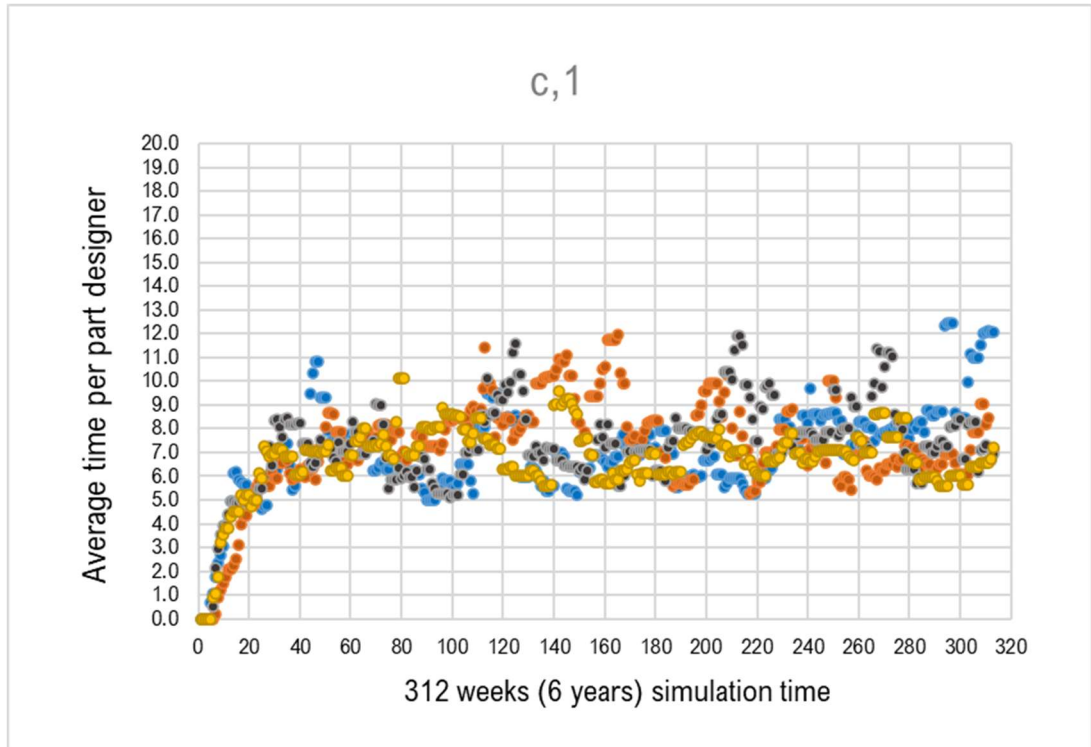


Figure 5.3.3.13 c,1 average time per part designer

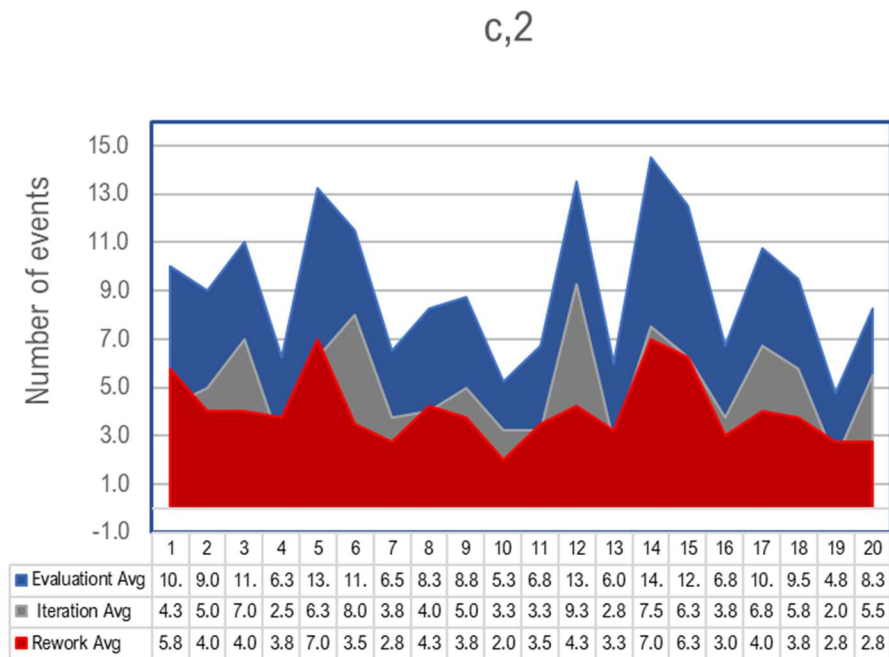


Figure 5.3.3.14 c,2 average number of events per run.

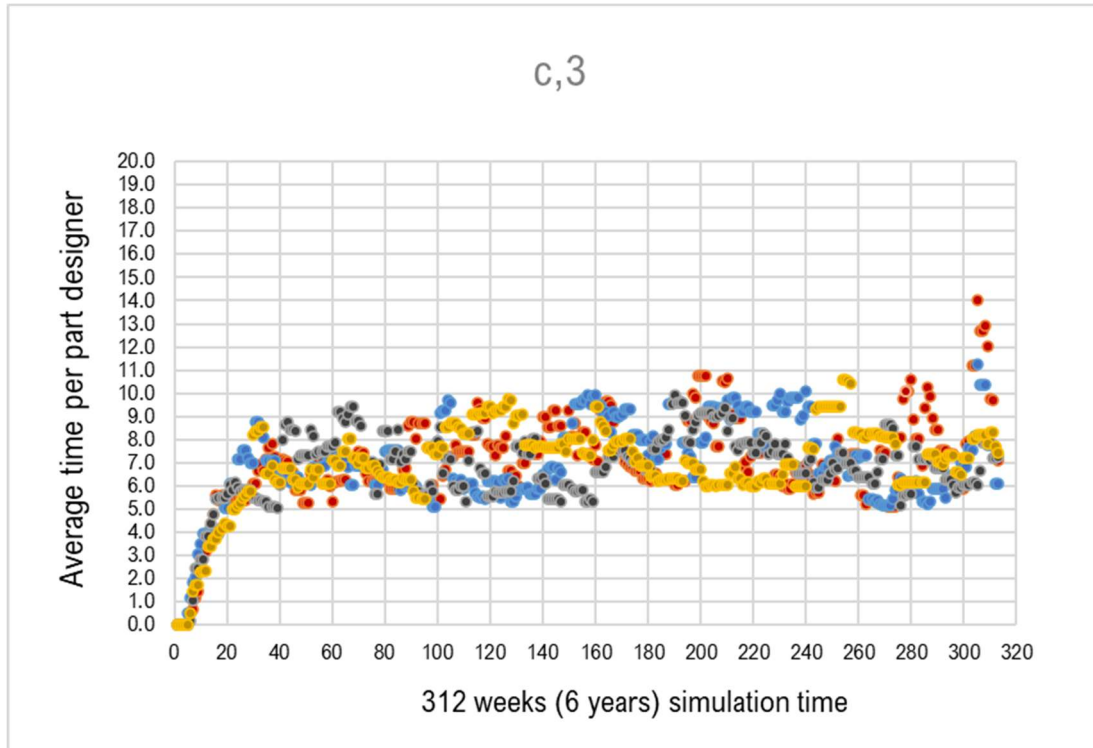


Figure 5.3.3.15 c,3 average time per part designer

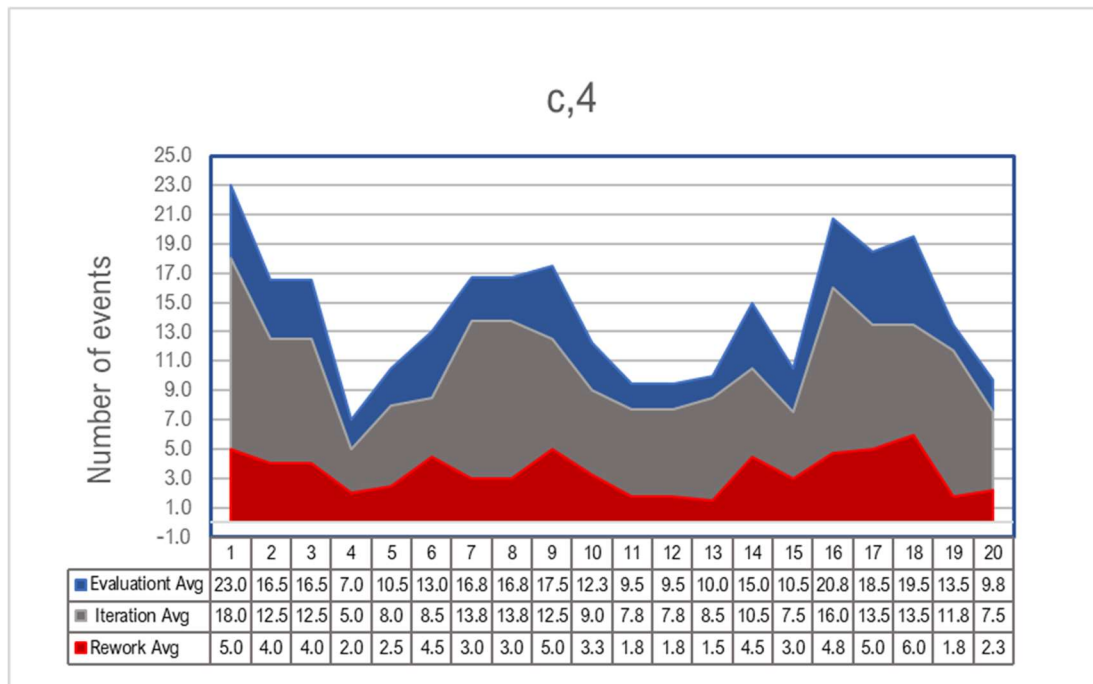


Figure 5.3.3.16 c,4 average number of events per run

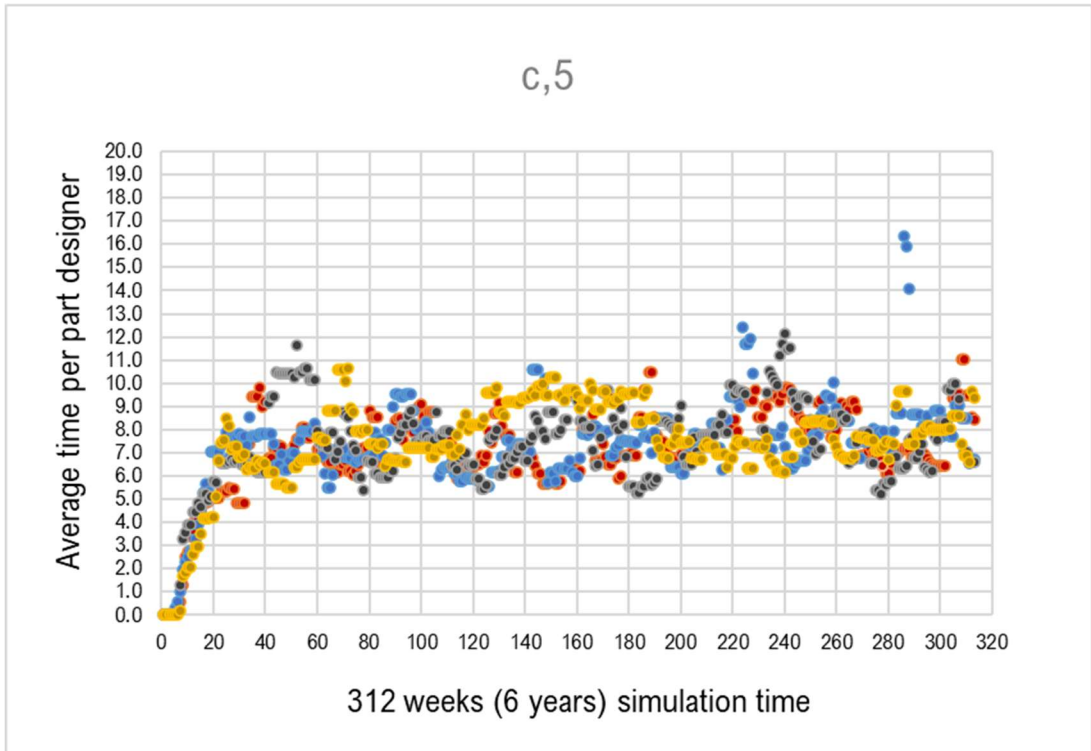


Figure 5.3.3.17 c,5 average time per part designer

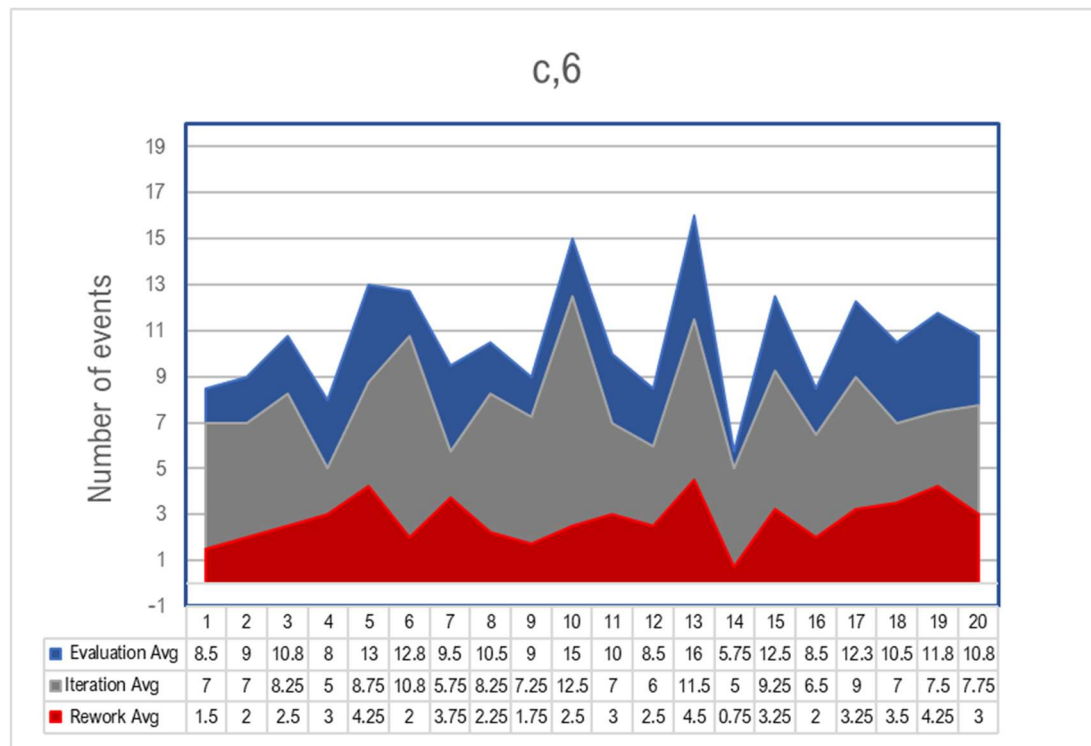


Figure 5.3.3.18 c6 average number of events per run

Design managers face a key challenge in balancing the dynamic nature of product development. Simulation modelling techniques offer a controlled environment for testing and analysing different scenarios, providing insights into how the system may respond to different inputs. This allows for the identification of strategies for managing complex real-world systems without incurring the risks or costs associated with real-life experiments.

By simulating scenarios where designers spend too much time gathering information or answering questions, delaying responses, project managers or design managers can identify potential risks that affect the efficiency of the overall design task. It has been found that moderate levels of communication lead to the most effective performance in engineering teams (Robinson, 2010). Conversely, insufficient and excessive communication levels are associated with performance decline (Patrashkova-Volzdoska et al., 2003). Therefore, a key challenge for design managers is to establish a balance between positive feedback loops in the form of design iteration and negative feedback loops in the form of avoidable rework (Tapia et al, 2021).

The analysis of experiments shows that moderate levels of information requirements, evaluations, and rework are not determined to produce stable outcomes in the process. Figures a,1, a,2, and a,3 show a dispersion in the points representing the times taken per individual designer or design team to conclude the design task. The lower average data is about 5 weeks, but at the top level, it is possible to observe points over 11 weeks and beyond, such as the point in figure a,3 between 140 and 160 weeks of the simulation time. On the other hand, despite the limited number of reworks reflected in figure a,6, the points in figure a,5 still reflect irregular patterns of time to complete the design tasks.

In a different note, experiments with four design requirements per year tend to show more regular patterns of time to complete the design task. In figure b,1, it is possible to observe that the concentration of data point of times to complete the tasks is allocated between five and nine weeks. Figure b3 presents a similar pattern, with less density between the ranges of five to ten. In both cases, there is a high rate of iteration and evaluation. Figure b,4 presents a peak of 24 events in run nine, with rework being low as expected.

An interesting turn in the trend of this set of data is figure b,6. Contrary to figures b,2 and b,4, there are no peaks in figure b,6, despite rework being set to be low in this scenario. In comparison with a,6, it is lower.

The experiment with six requirements per year presents a particular behaviour. The density of the points in the scatter graph c,3 shows similarities with c,5. However, comparing the data from b,6 with c,4, the iteration and evaluations are much higher.

5.4 Documentation

The documentation of the conceptual framework implementation will be allocated in the online open access research repository figshare.

Figshare <https://figshare.com/account/home#/data>.

Contents:

- Simulation documentation.
- Simulation files.
- Conceptual model diagrams.
- Simulation results data, spreadsheet files.

5.5 Summary

The Action Design Research (ADR) development process consisted of developing several BIE cycles of simulation models, starting with the discrete events simulation method, observing how to design iteration and rework, and the social interactions within and across design teams influenced the progression and quality of design tasks. In the discrete events simulation models, changes in the inputs are reflected directly in the time taken for an entity to pass through the simulation system until the end. In the resulting discrete event simulation model, iterations are selected feedback loops randomly happening with the part design process. In contrast, rework loops span the design processes of multiple parts and are modelled as delays.

During the second part of the BIE cycle, the development of the agent-based model focused on the characteristics of individual agents (designer or design teams). The agent-based simulation model considers the design activity from a socio-technical perspective that involves the communication behaviours of design teams, the influence of social interactions in the

process and the product architecture identified as the critical element for developing realistic simulations. Capturing iterations and rework in the form of feedback loops because of the interactions between the agents when they communicate.

In the third part of the BIE cycle, the resulting hybrid simulation model integrates the agent-based models with the discrete event model. In this model, time task completion is determined for the agent-based structures, which are also influenced by the design teams' information and communication behaviours. A relevant contribution of the third BIE cycle is identifying interaction points between simulation models.

5.5.1 Scrum sprint simulations





The implementation of the simulation model for the Scrum Sprint agile process, developed from the model formalization stage, is structured with consideration of actors, relationships, interactions, behaviours, and states. It is important to highlight that not all identified system components will be included in the simulation model. The Sprint methodology primarily focuses on design development teams, with different products of the product architecture integrated into the product backlog as priority tasks. The actions of the Product Owner and Scrum Master are represented as assumptions, such as in the acceptance or non-acceptance of finished tasks. Table 5.5.1 provides a summarized representation of actors, relationships, interactions, behaviours, and states.

Table 5.5.1 Summary of actors, relationships, interactions, and states.

Sprint Development Team	Design team for the handlebar assembly	Sprint Planning	Updates priorities	Reviewing
		Sprint Iteration	Works in the design task	Designing
		Sprint review	Asses product increments	Assesing
		Sprint retrospective	Feedback in process and product incrfement	Reviewing
		Defintion of done	Determine acceptance criteria	Accepted/ not accepted

The result of the preceding stage is the definition of potential parameters utilized for the simulation. Table 5.5.2 illustrates the possible parameters employed to configure the hybrid simulation model, encompassing the parameters utilized in the state chart constructs that represent the sprint development team and the stage gates development process.

Table 5.5.2 Simulation initial parameters. Description of blocks form Anylogic 8

Elements	Block	Description	Parameter	Initial value
Scrum End		Event is the simplest way to schedule some action in the model	Timeout, weeks	4.5 Weeks
Sprint planning		State	N/A	
Transition		A transition denotes a switch from one state to another	Condition	True
Daily Scrums		State represents a location of control with a particular set of reactions to conditions and/or events.	N/A	
Transition		Time for the Daily scrum meeting	Timeout, minutes	15
Sprint Development		State	N/A	
Transition		Time for the Sprint development	Timeout, weeks	1,2,4,5
Sprint Review		State	N/A	
Transition		Time for the Sprint review meeting	Timeout, hours	3 hours
Sprint retrospective		State	N/A	
Transition		Time for the scrum end controlled for the events construct	Timeout, weeks	4.5 Weeks
Done		State	N/A	
Transition		Time for decision	Timeout, hours	1 hour
Definition of done		Branch	Condition	Random True (.5)
Complete		State	N/A	
Not Complete		State	N/A	
Design request		Generates agents	Rate, per month	6-12 per year
Queue		Works as a buffer for agents	Maximum capacity	N/A
Wait		Stores agents inside and supports their manual retrieval	Free all()	N/A
Finished design		Routes the incoming agents to one of the two output ports depending on (probabilistic or deterministic) condition.	Boolean	If condition its true
Finished design		Disposes agents	N/A	End of the process

The implementation of the simulation model created in Anylogic 8 follows the Scrum Sprint narrative introduced in Section 4.6.1.5. It begins with a Sprint Planning activity, followed by Daily Scrums, Sprint Development, Sprint Review, and Sprint Retrospective, all incorporated within a composite state. The process culminates with a "done" state, which is then assessed as either completed or not.

The discrete events model injects agents into the simulation. The entire process starts when the system utilizes the discrete events block "wait" to pause the discrete events process until the agent-based model completes the Sprint and a decision regarding its completion status is made. If the agent-based model's decision is "not completed," the decision block within the discrete events sends the agent back to the queue block for reintroduction into the system, creating a backlog. In this study, we've identified this as a cross-gate iteration.

The simulation uses three parameters to evaluate different scenarios:

- 1) Quantity of requirements per year, which simulates the team's workload at varying levels: 6, 8, and 12 requirements per year.
- 2) The Sprint time parameter, which simulates different Sprint durations ranging from 1, 2, and 4.5 weeks.
- 3) The Scrum end parameter, which simulates the extension of the Scrum parameter. In the simulation, it is set to a length of 4.5 weeks, as suggested in Agile Scrum literature.

The Scrum Sprint simulation model, integrating both the agent-based model and the discrete events model, is depicted in the following diagram.

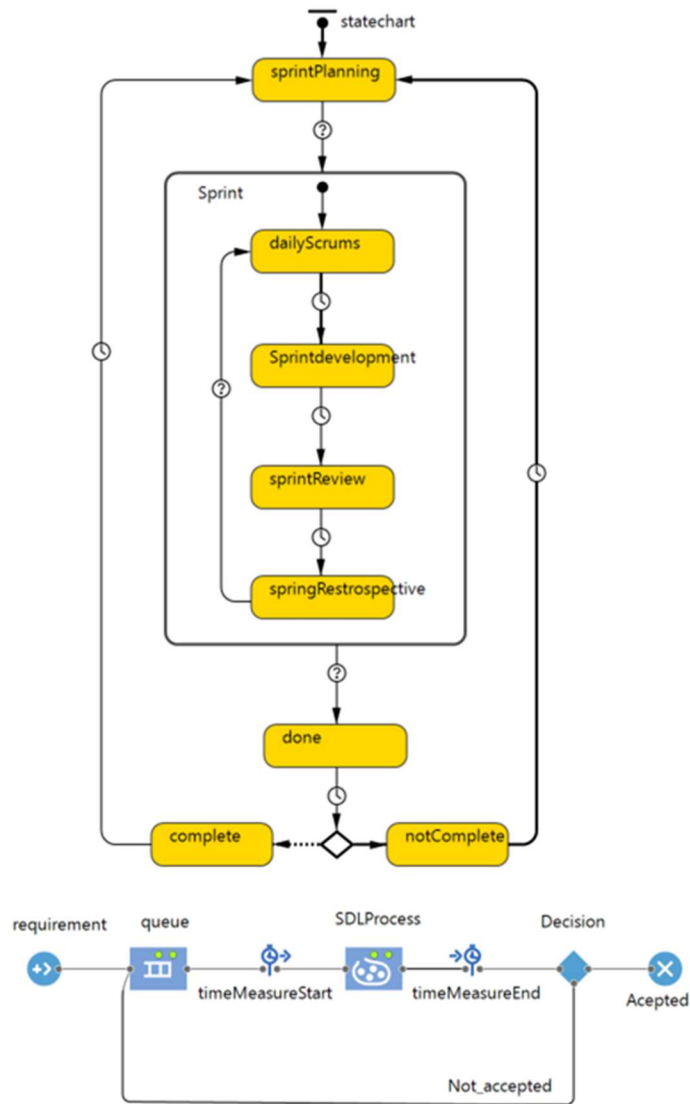


Figure 5.5.1 Simulation model diagram made in Anylogic 8.

A design of experiments for testing sets of parameters and graphs to represent the results is presented in the following section. The experimental design consists of three scenarios, each including three experiments. Each experiment varies the number of requirements per year, and each scenario adjusts the Sprint duration in weeks. All scenarios use a random seed setting and run for 104 weeks (two years).

Table 5.5.3 Experiments design specification.

INPUTS								
Scenario	Experiment No.	Requirement projects per year	Sprint time in days/weeks	Scrum end in weeks	Complete /Not complete	Number of simulation runs	Randomness	Time of simulation run in weeks
1	1	6	4.5	4.5	50%	10	Random seed	104
	2	8	4.5	4.5				
	3	12	4.5	4.5				
2	1	6	2	4.5	50%	10	Random seed	104
	2	8	2	4.5				
	3	12	2	4.5				
3	1	6	1	4.5	50%	10	Random seed	104
	2	8	1	4.5				
	3	12	1	4.5				

In following section figures 5.5.2, 5.5.3, 5.5.4 display graphs with the results for the average time it takes for the Sprint development team to complete tasks within the Sprint event, using the parameters of the three different scenarios. The mean column reports the time to complete the sprint in the three scenarios.

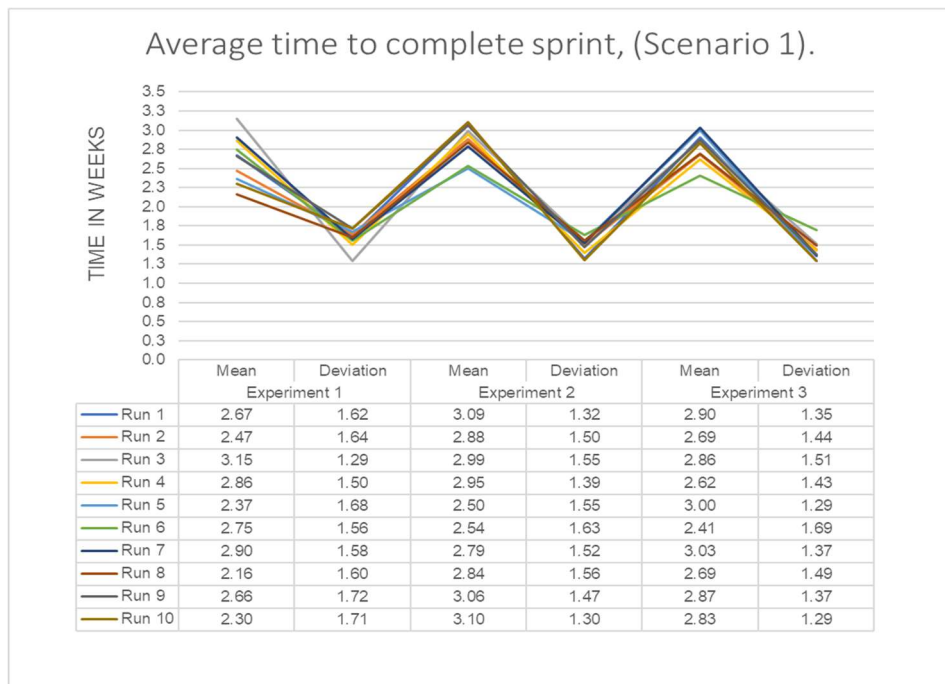


Figure 5.5.2 Average time to complete a sprint (Scenario 1)

Graph 5.5.2 summarizes the results of 10 runs with the parameters of scenario one, experiment one, considering a requirement of 6 projects per year, 4.5 days per sprint and 4.5 weeks for the length of the sprint event.

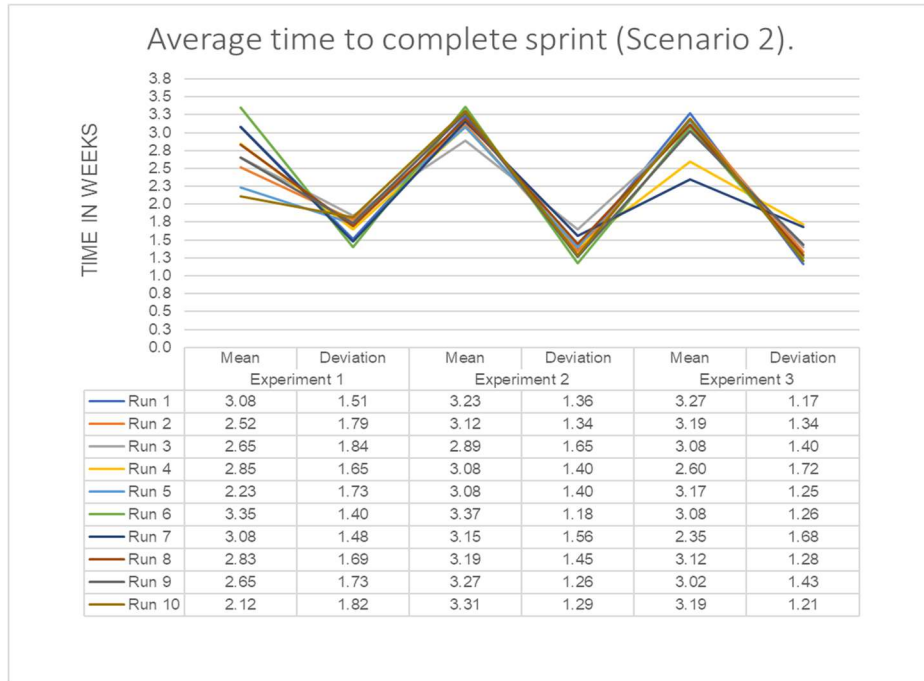


Figure 5.5.3 Average time to complete a sprint (Scenario 2)

Figure 5.5.3 depicts the graphs shown results of experiments of scenario two, with eight requirements per year.

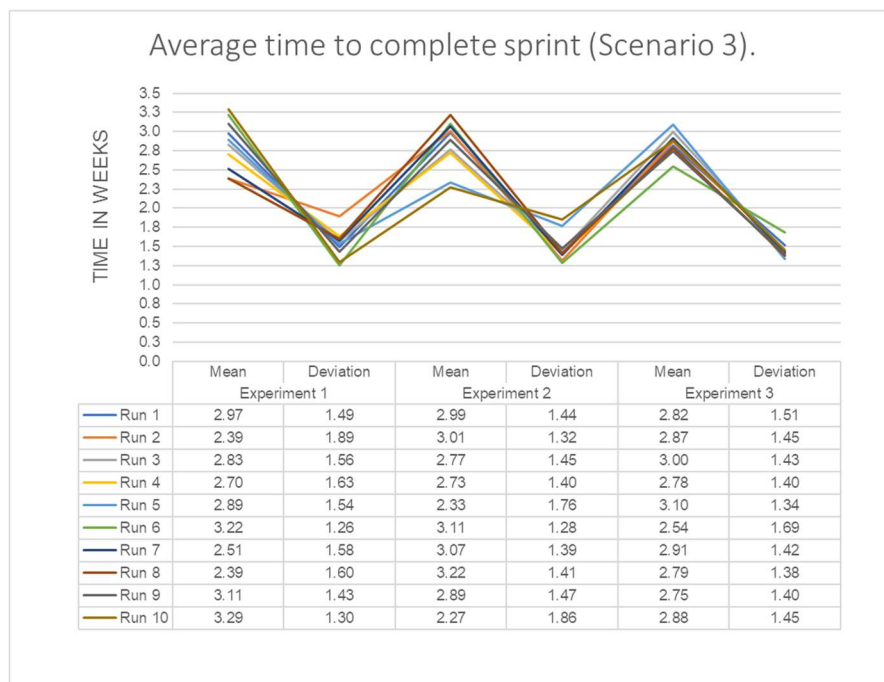


Figure 5.5.4 Average time to complete a sprint (Scenario 3)

In figure 5.5.4 the results of scenario 3 are presented. The analysis of the results of the three scenarios shows that time in scrum sprint events tends to be homogeneous with a range of variations between 2.1 to 3.4 weeks per event despite the number of requirements per year introduced in the experiments.

The following section presents graphs 5.5.5, 5.5.6, and 5.5.7 illustrating the results of the scenario experiments, evaluating the average number of requirements among the accepted and not accepted design jobs. In the graphs each column presents the data from experiments one (blue), two (orange) and three (grey). The graphs show the average of not accepted, the average of accepted designs and the average of requirements per year.

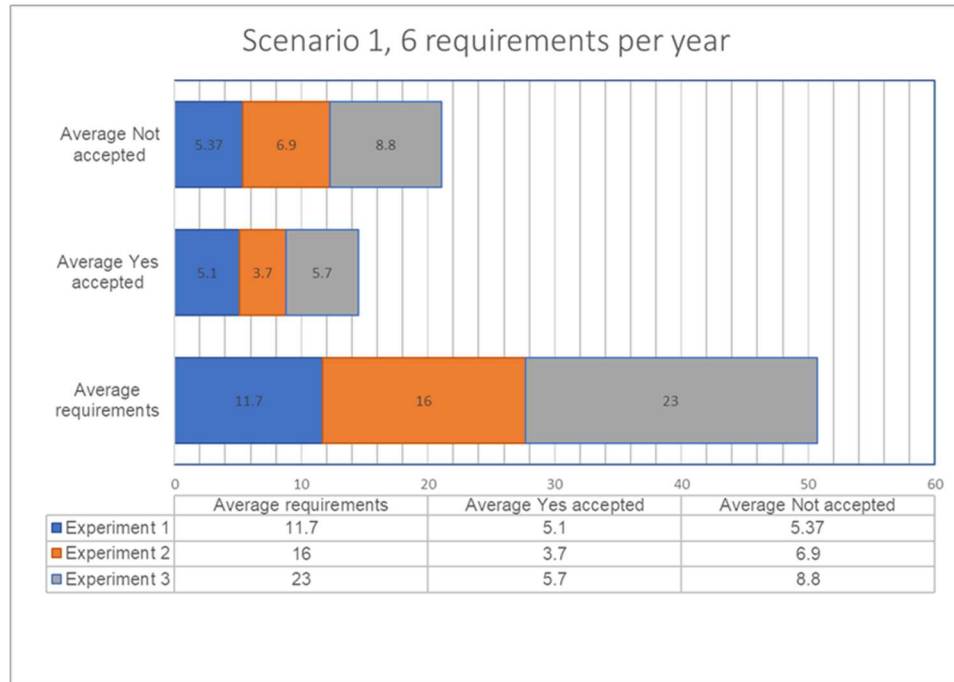


Figure 5.5.5 Average of accepted and not accepted designs (Scenario 1)

Figure 5.5.5 shows the average of accepted and not accepted jobs using parameters from scenario one. The simulation experiment runs for 104 weeks.

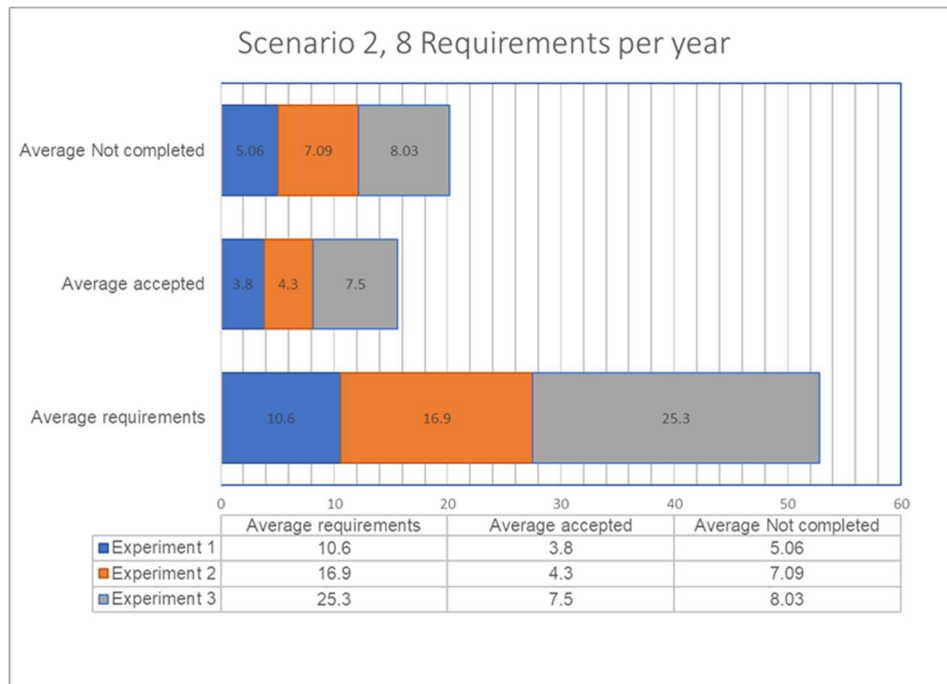


Figure 5.5.6 Average of accepted and not accepted designs (Scenario 2)

Figure 5.5.6 presents the results of average accepted and not accepted designs of experiments in scenario two.

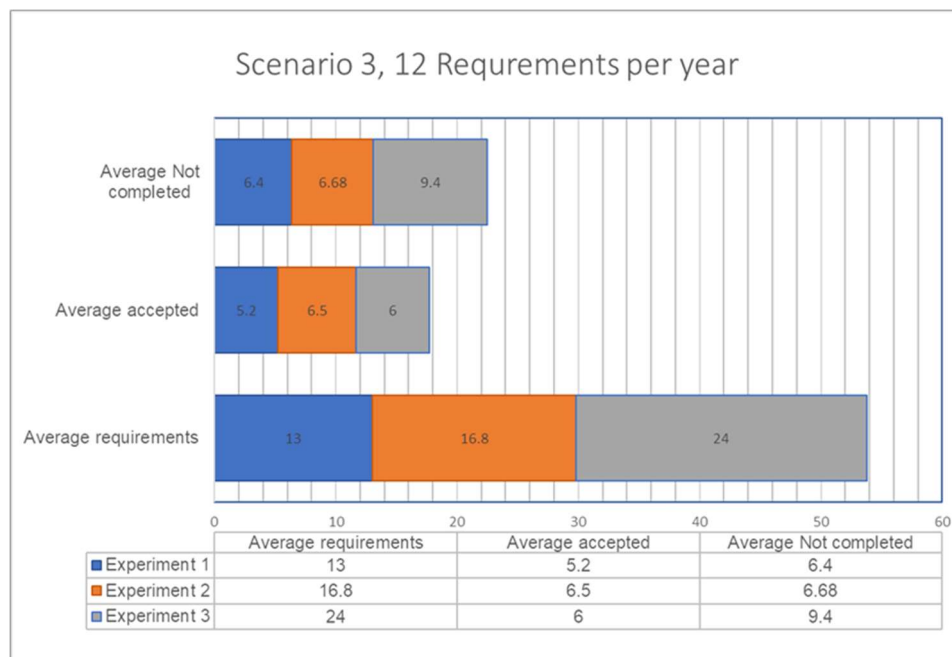


Figure 5.5.7 Average of accepted and not accepted designs (Scenario 3)

The result of this analysis of the data collected in the graphs 5.5.5, 5.5.6 and 5.5.7 shows that amount of not accepted designs is not affected by the

number of requirements, in the graphs the amount of not accepted designs from 6.4 to 9.4 is slightly smaller than the number of accepted designs, from 5.2 to 6 designs per year.

The next set of graphs, figures 5.5.8, 5.5.9 and 5.5.10, show the data regarding the number of sprints among the design jobs done, each column represents a run and the experiments are paired the design jobs done and the total of sprints, for example run one have “Experiment 3 jobs done” paired with “Experiment 3 Total Sprints”.

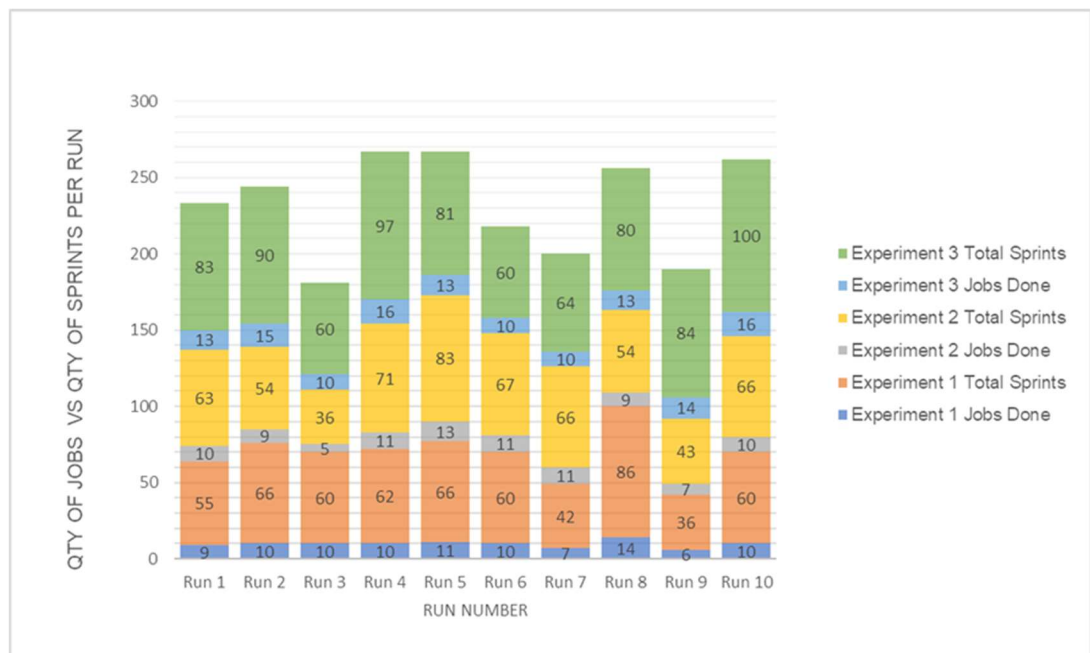


Figure 5.5.8 Average design jobs vs Sprints per run (Scenario 1)

The graph in figure 5.5.8, shows the results of experiments in scenario one.

In the experiment one the short time given to the sprints is noticeable. In a range of 36 to 66 in the orange sections, from 36 to 83 in the yellow sections and from 60 to 100 in the green sections.

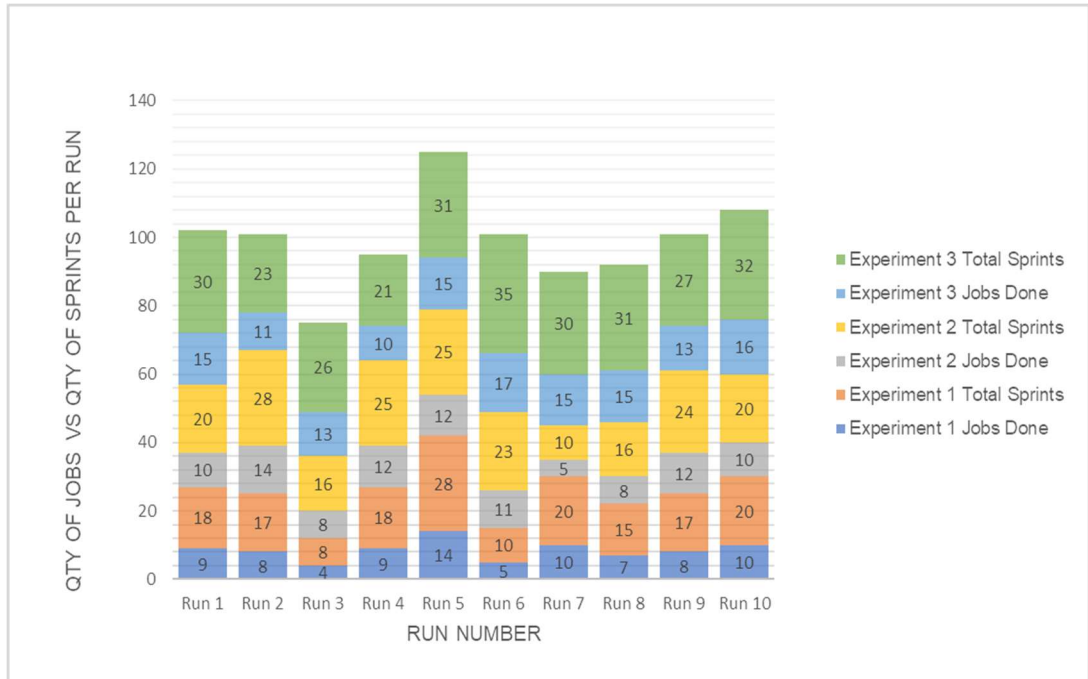


Figure 5.5.9 Average Design jobs vs Sprints per run (Scenario 2)

The graph in figure 5.5. 9, shows the results of experiments in scenario number two, which show a more balanced number of sprints among the design jobs done, for example run number one experiment one shows 18 sprints per 9 design jobs.

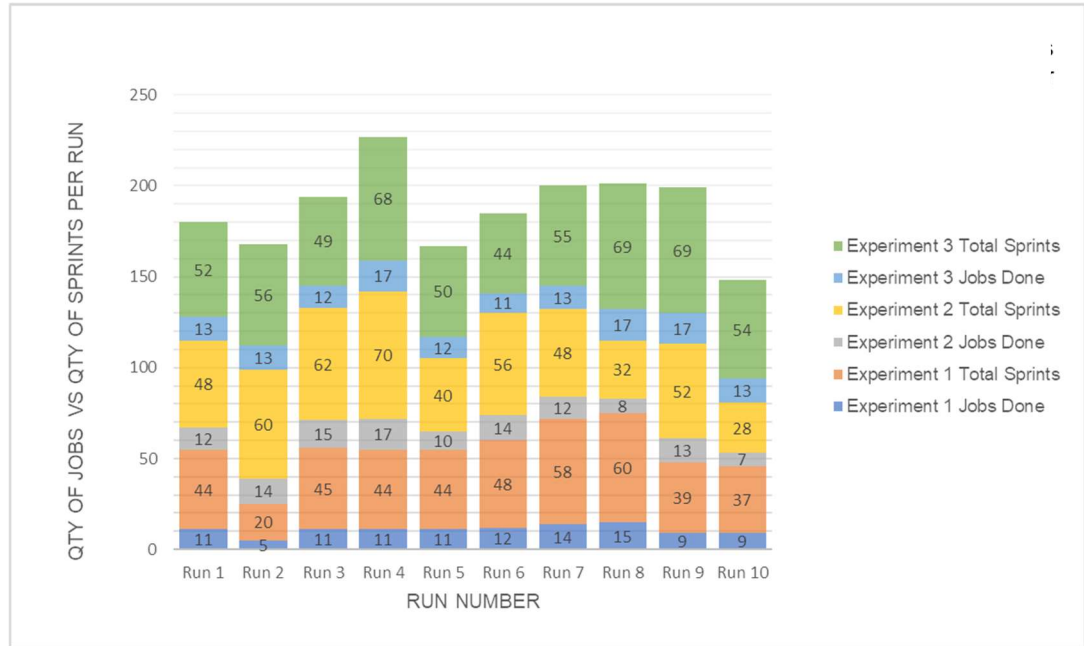


Figure 5.5.10 Average Design jobs vs Sprints per run (Scenario 3)

The graph presented in figure 5.5.10 present the data from the scenario 3.

Section 4.6.1.5 demonstrates that the conceptual framework possesses the flexibility to be adapted for simulating Agile sprints. By repopulating the conceptual framework with Scrum-sprint constructs, the model successfully integrates the elements, relations, agent types, behaviours, and states specific to agile methodologies.

The model allocates Scrum agents to their corresponding components, it also introduces interaction points, such as 'Process start,' 'Sprint Review,' and 'Definition of Done,' which demonstrated the model's ability to capture and represent the complex, dynamic nature of agile processes. This adaptability highlights the conceptual model's capacity to comprehensively analyse and simulate agile sprints by considering the interplays between people, processes, and technology within these socio-technical systems.

At this stage, the simulation model can produce simulations that visualize the time taken by the sprint development team to complete a task, the number of tasks compared to the total number of sprints during the Scrum event period, and the count of requirements among the accepted and non-accepted sprints. The variations in the selected parameters, such as the quantity of requirements and sprint duration in the design experiments, do

not seem to significantly influence the number of non-accepted jobs. In all three scenarios, the number of non-accepted jobs is higher than the number of accepted ones.

On the other hand, the simulation model suggests that one of the main problems of the agile sprint method is that tends to produce unnecessary iterations, the results of the experiments show that a larger number of sprint iterations, produce in proportion the same amount of non-accepted jobs that a shorter or a regular amount. To be able to produce more realistic simulations, it is necessary to identify how the produced design increments during the sprints are identified and assessed and how these are produced during the sprint iterations.

Future works suggested by this study include:

- 1) Investigating the parameters defining 'the definition of done' to better understand how Scrum Sprint Agile methods evaluate the 'increments' in the design.
- 2) Since this framework adopts a socio-technical approach, the study needs to gain an understanding of the communication behaviours and information processing of the teams working under Scrum Sprint Agile.

Chapter 6: Conclusions

In the engineering design literature, design iterations are defined as revisiting an already finished task to add more information (Ulrich and Eppinger, 2012). Iterations are seen as learning cycles (Meboldt et al., 2013) that contribute to a better understanding of the design problem and solutions (Eckert et al., 2014). Design iterations can be seen also as cycles of learning and knowledge accumulation that lead to creativity in what is known as the co-evolution of problem-solution space suggested by Dorst and Cross (2001). During these cycles of learning and creativity, designers develop and refine the formulation of a problem and ideas for solution in a constant cycle of analysis, synthesis, evaluation, and decision making (Chusilp and Jin, 2006) at individual or small teams level (Wynn and Eckert, 2017). On a different note, the iterations performed by teams, due to new information arriving or because previous iterations (Chusilp and Jin, 2006), are used to generate meaningful data to define and refine a design solution towards a desired state (Costa and Sobek, 2003). These iterations are influenced by technology, communications and new design solutions when design teams, customers and suppliers iterate to converge on a practical design solution (Eckert et al., 2014).

Rework is defined as revisiting an activity at the same abstraction level on the same part of a design to correct errors made earlier (Costa and Sobek, 2003). Drivers for rework include a prior decision that was found to be defective (Kennedy et al., 2014), or a change due to initially imperfect information or changes in requirements (Smith, R., P. and Eppinger, 1997b; Taylor and Ford, 2006b). Causes of rework during project execution are often related to project complexity (Cho and Eppinger, 2001) and information evolution and completeness (Dullen et al., 2019).

These reflections led to the conclusion that iterations and rework are forms feedback loops. With different impacts on the product and process, iterations seek to improve and evolve a design while rework aims to correct and control. Meboldt et al. (2013) assert that In-stage Iterations are feedback loops within a development stage that have limited effect on the previous gates' decisions and lead to product maturity (Krehmer et al., 2009). Conversely, cross-gate iterations require changes to decisions made in earlier stages or trigger issues at the end of the product development process; both are expensive in time and cost.

On a different note, dynamical systems are those whose behaviour changes over time. Feedback loops result in a situations where two or more dynamic systems are interrelated, influencing each other in a strongly coupled circular manner (Åström and Murray, 2021). In social and organisational theory, feedback loops are essential to understanding relationships between complex social systems (Tsoukas and e Cunha, 2017). The term feedback loop is used to define activities aimed to reduce a gap between a perceived and future state of a system and to explain how a process's outputs influence its input directly or indirectly at some point in time. Feedback loops are associated with the evolution of a system over time when they are related to control and stabilisation and with the improvement or with the decline of a process or behaviour when they are associated with virtuous and vicious circles (Masuch, 1985). Vicious circles are feedback loops that turn a negative situation into worse. On the contrary, virtuous circles are feedback loops that improve a good condition (Tsoukas and e Cunha, 2017).

The key research challenge of this study was to find ways of identifying the interplays between positive and negative feedback loops in product development systems with a view to providing tools that allow managers to balance positive feedback iterations and negative feedback loops, in the form of avoidable rework. A conceptual framework for simulating feedback loops in engineering design was established in response. This framework couples a generic process model of product development with the product architecture of the designed product. The conceptual framework combines two simulation methods to form a hybrid simulation model. An agent-based simulation model reflects design activities, communication and design iterations, and a discrete events model reflects the product architecture and stage gates influencing the process development structure and capturing the rework feedback loops.

6.1 Research Contribution.

The contributions of this thesis are described in the following sections:

Objective No 1: *To identify key technical and social aspects of product development processes for the simulation of feedback loops in engineering design.*

The following aspects were identified.

- The product development process is a sequential set of development activities where engineering teams carry out design work, separated by stage gates where go/no go decisions are made. Together, these form the process workflow (Shepherd and Ahmed, 2000; Montagna and Cantamessa, 2017). Meboldt et al., (2013) suggest that two types of iteration are present during the stage gated development process: in-stage iterations, that do not impact decisions made in previous stages; and cross-gate iterations where decisions affect decisions made in previous gates, so impacting project time and cost (Meboldt et al., 2013).
- Design iteration is an in-stage iteration, which occurs within each development stage, where design activities are carried out. They tend to improve the quality of the design within a design stage. In these iterative cycles designers revisit and re-evaluate previous design decisions (Wynn and Eckert, 2017) resulting in new activities and feedback loops.
- Rework is a form of cross-gate iteration. It affects decisions made in previous stages so affecting project time and cost. Rework is a result of information dynamics in new product development processes and is, caused by inadequacy of information due to changes in requirements, poor decisions, defective outputs or changes in implementation that alter work previously done (Smith, R., P. and Eppinger, 1997a).
- Product architectures are described by their elements and relationships (McKay et al., 2016). They define system boundaries, including functional and physical configurations (Jankovic and Eckert, 2016). Product architectures, inform the structures of the product development process and are used to establish the development tasks and design activities (McKay et al., 2022; Jankovic and Eckert, 2016) that result in design descriptions and including shape definitions and material specifications (McKay et al., 2016).
- The product architecture also influences technical communications and interactions within design teams (Clarkson and Eckert, 2010). During design activity, engineering designers and stakeholders interact to find a design solution (Montagna and Cantamessa, 2017). Communications, negotiation, and coordination mechanisms are determinant for the outcome and progress of the design work (Hoegl and Weinkauff, 2005; Maier, A.M. et al., 2007).

Given these aspects, the engineering design processes within product development can be regarded as complex information-processing activities, consisting of creating, transferring, or disseminating information (King, 1994) directed by the decisions made by individuals in design teams (Wallace and Ahmed, 2003). Engineering designers spend 24% of their working time in activities (Marsh, 1997), searching for information, identifying relevant sources, accessing and acquiring information from those information sources, processing and analysing the obtained information, and finalising the search process (Meho and Tibbo, 2003). This research included all four of information-seeking stages: Requesting, Answering, Receiving, and Evaluating information.

Objective No 2. *To identify critical characteristics of feedback loops that influence the performance of product development processes.*

The performance of product development processes is typically assessed through four key performance indicators.

-Time: The time taken to render a design from a set of requirements to the delivery of the product.

-Cost: The resources used in designing and delivering products to the market. This typically includes both financial resources and staff time used, which can impact monetary costs.

-Quality: The extent to which the given design fulfils stakeholders' expectations and the expected value.

-Responsiveness: The ability of an organisation to respond to change.

In the engineering design literature, it has been established that design iteration adds time and cost to the design activity. For example, Costa (2004) asserts that "iterations shape the outcomes of the design in terms of cost, time and quality", and continues arguing that iterative approaches tend to increase development time and cost. On the other hand, design iterations are learning cycles that contribute to knowledge acquisition and the mitigation of uncertainty and ambiguity. Iteration feedback loops improve quality by the systematic exploration the design problems leading to an efficient design problem-solution finding process (Le, H.N. et al., 2010). Although design iteration adds time to the design activity, it has the potential to improve the resulting design by contributing to design quality (Tapia et al.,

2021). On the other hand, rework iterations consume time (Arundachawat et al., 2009) and therefore affect project duration and cost (Tapia et al., 2021). Rework iteration results from changes in requirements or repetitions of tasks due to initially imperfect information (Smith and Eppinger, 1997), producing adverse effects and the need to redo tasks, impacting other stages of the development process.

In this study the critical characteristics of feedback loops that influence the performance of the product development processes are related to quality, accuracy, and timely information. When the information is adequate, accurate and produced on time within design activities, these iterative cycles are positive iteration feedback loops contributing to the quality of the design. On the contrary, when new information needs to be more accurate or adequate, requiring modifications on previous activities from earlier phases considered already finished. These are negative rework feedback loops affecting project performance and design quality.

Objective No.3, *To design and develop a conceptual framework for simulating feedback loops in engineering design that incorporates the identified critical elements for the implementation of more realistic simulations of product development processes.*

The conceptual framework for simulating feedback loops in engineering design, introduced in Chapter 4 uses linear logic of the traditional stage gate product development process model to establish a chronological structure with gates where decisions to proceed or not are made. The architecture of the product being designed is used to identify the parts to be designed, leading to the identification of the required agents (designers or design teams) to perform the engineering design processes within each stage of the development process. The framework couples social characteristics of engineering design activities with processes (consisting in designing, communicating, processing and evaluating information) and iteration feedback loops. The communication patterns identified are related to asking and answering questions and the information patterns are identified as gathering information, processing information and evaluating information. The feedback loops identified are in-stage iterations, which do not impact decisions made in a previous stage; and cross-gate iterations where decisions affect decisions made in previous gates.

The chronological structure of the traditional product development process model is a relevant feature of the conceptual framework because it provides an organisation of the tasks and events in the order in which they occur in the process workflow. The relevance of the product architecture in the conceptual framework it enables the identification of the parts and relationships of the product to be designed so providing the identification of physical elements, these elements in turn determine the necessary agents and is useful in the identification of the limits of the system being simulated.

The product development chronological structure and the product architecture makes possible the establishment of workflow as the sequence of steps necessary to complete the specific tasks to complete a design. The social interactions of the designers and design teams communicating and processing and evaluating information, resulting from engineering design tasks activities, is a determinant for the performance of the product development process.

In summary the research contribution of the conceptual framework for simulating feedback loops in engineering (presented in Figure 4.4.2) is that such a model enables the analysis of both intangible aspects, (communication and information processes), as well as tangible aspects (product architecture). In this way a comprehensive view of the system's performance (Škec et al., 2017) becomes possible

The application of the conceptual framework for the simulation of feedback loops in engineering design will enable the development of realistic simulations with a comprehensive view of the engineering design process, considering the technical aspects of the product development, such as the product architecture of the product, the linear logic of the product development process, and social aspects of design teams' communication patterns and feedback loops.

Objective No. 4, *To implement an engineering design process case study for use in validating the framework.*

An engineering design case study (introduced in Chapter 3 and used to validate the conceptual framework in Chapter 5), was used to validate the conceptual framework The case study focused on a bicycle handlebar assembly, which consists of a handlebar, brake lever, gear change lever, and the handlebar assembly itself. The design architecture of the handlebar

assembly, with three independent parts integrated in a final assembly, was used to identify the general configuration of the simulation model, with each part of the product architecture being designed by a different design agent. The conceptual framework guided the implementation of the agents performing design tasks of analysis, synthesis and evaluation and design activities that include, information processing, and communication activities. The structuration of the agents used the outcomes from the Section 5.1 implementation to identify, behaviours, interactions, and states. The design agents are integrated into the product development process structure, which is represented as a sequence of operations performed across activities (Borshchev, 2013) informing the workflow and the discrete event simulation model (shown in Figure 5.1.2.2).

In the simulation model, the interaction points between the simulation models are identified as pairs of inputs/outputs of information exchange (Chahal, 2010). Each interaction is "captured by" or "influenced by" each of the simulation models (Mykoniatis and Angelopoulou, 2020). The simulation case study, presented in Chapter 5, captures the interactions between different teams and processes involved in designing the bicycle handlebar assembly using both a discrete events simulation model and an agent-based model.

Objective No.5, *To consider how such simulation models might be used to inform the management of product development systems.*

The research contribution includes two features of the conceptual framework for simulating feedback loops in engineering design. The first feature is the use of parameters that influence social interactions. These parameters include factors like asking and answering questions as well the time it takes to answer a question. By incorporating these parameters into the simulation, project managers or design managers are able to simulate the risks of information misuse during the design process. Specifically, these parameters can be used to simulate scenarios where too much time is spend gathering information, answering questions from other designers or delaying responses to information requests. Additionally, the simulation can also account for scenarios where designers have to ask for clarifications or expansions on information, by simulating these scenarios project managers or design manages are able to identify potential risks affecting the efficiency of the overall design task. Moderate levels of communication lead to most effective performance in engineering teams (Robinson, 2010) conversely

insufficient and excessive communication. levels are associated with performance decline (Patrashkova-Volzdoska et al., 2003). Design managers or project managers can counteract these effects by running simulation experiments to identify features that define good practices or an information policy, regarding information and communications between project participants.

The second contribution is the incorporation of the product architecture as a key feature in the construction of the simulation model. Product architectures inform the structure of the product development process and are used to establish development tasks and design activities (McKay et al., 2022; Jankovic and Eckert, 2016). As a result, they also influence technical communication and interactions among design teams (Clarkson and Eckert, 2010). The conceptual framework allows managers to experiment with alternative product architectures and management strategies and, through computer simulations, gain insights on how this impact the product development performance. A key challenge for design managers lies in deciding how to define a product architecture that will result on the best performing product development process. These techniques offer a controlled environment for testing and analysing different scenarios, providing insights into how the system will respond to different inputs. This allows the identification of strategies for managing real-world complex systems without incurring the risks or costs associated with real-life experiments.

The conceptual framework for simulating feedback loops in engineering design, leverages the experimental characteristics of simulation modelling, using a discrete-event modelling approach to reflect the product architecture and its influence on the product development process structure, including potential rework feedback loops. Additionally, an agent-based model reflects the social facets of design activity and communication behaviours within design teams, including design iteration.

6.2 Research Limitations.

This study represents a first step in developing conceptual frameworks for the simulating feedback loops in engineering design. However, two limitations were identified. First the absence of a real-world engineering design case study that includes process information that allow the

experimentation with different configurations of the development process and product architecture. Second, as a consequence of the absence of a real case study there were limited opportunities to use traditional validation methods for simulation.

At this stage of the research the simulation model can represent feedback iterations and rework happening during the design process as a group of simulation constructs, influencing the system performance. However, there is a need for an engineering design case that include process information, which allows the identification of system elements and creates the opportunities to experiment with alternative configurations to observe not only the intangible aspects but also include tangible aspects of the development process.

On the other hand, Traditional validation methods for standalone simulations are concerned with whether the model is an accurate representation of the real-world system by comparing experimental results with real-world data (Sargent, 2020). However, Nikolic and Lukszo (2013) assert that is not possible to compare a computed behaviour to real system behaviour if there is not real system available for comparison. They assert that the value of complex systems simulations is its ability to explain the system possible operation or potential states (Nikolic and Lukszo, 2013). The use of a synthetic engineering design case study led to several assumptions of the system configuration and performance that is difficult to validate against a real-world system.

6.3 Future Work.

This section outlines work that could be carried out to cope with the mentioned limitations identified in Section 6.2 and, more widely, to expand across the simulation of feedback loops in engineering design in the short, medium, and long term.

In the short term, Nikolic and Lukszo (2013) suggest a strategy to validate the model by literature comparison, which involves identifying those studies that reached similar conclusions, This could include the implementation of experiments in the case of this research to explore how early design decisions in product architecture, may impact, the performance of the product development process. Such studies rather than focusing on replicating the exact outputs of other studies would observe the general

outcomes and recommendations. The model validity increases when the model recommendations are compatible with the available theory and case studies (Nikolic and Lukszo, 2013).

In the medium term. Qualitative research studies to identify constitutive elements of the product development systems in real-world organisations, using structured questionnaires and workshops. This would enable the validation of the model assumptions, mechanisms, and outcomes, in the form of characteristics of design activities process structures, and product architectures among academic and practitioner experts.

In the long term, For the use in design management, end user interface for non-computer sciences background individuals is need. This would allow project managers and design managers the experimentation with different parameters, i.e., configurations of systems architectures, designers' behaviours, project constraints, and feedback loop iterations. to evaluate different system configurations and find the best possible arrangement for a particular project or product.

List of References

- Artmann, C. 2009. Literature Review. In: Artmann, C. ed. *The Value of Information Updating in New Product Development*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp.1-31.
- Borshchev, A. 2013. Multi-method modelling. In: *Proceedings of the 2013 Winter Simulation Conference: Simulation: Making Decisions in a Complex World*, pp.4089-4100.
- Borshchev, A. and Filippov, A. 2004. From system dynamics and discrete event to practical agent-based modeling: reasons, techniques, tools. In: *Proceedings of the 22nd international conference of the system dynamics society*: Citeseer.
- Browning, T.R., Fricke, E. and Negele, H. 2006. Key concepts in modeling product development processes. *Systems Engineering*. **9**(2), pp.104-128.
- Chiva-Gomez, R. 2004. Repercussions of complex adaptive systems on product design management. *Technovation*. **24**(9), pp.707-711.
- Chusilp, P. and Jin, Y. 2006. Impact of mental iteration on concept generation. *Journal of Mechanical Design*. **128**(1), pp.14-25.
- Clegg, C.W., Robinson, M.A., Davis, M.C., Bolton, L.E., Pieniazek, R.L. and McKay, A. 2017. Applying organizational psychology as a design science: A method for predicting malfunctions in socio-technical systems (PreMiSTS). *Design Science*. **3**.
- Costa, R. and Sobek, I.D.K. 2003. Iteration in engineering design: inherent and unavoidable or product of choices made? In: *ASME 2003 International design engineering technical conferences and Computers and information in engineering conference*: American Society of Mechanical Engineers, pp.669-674.
- de Weerd-Nederhof, P.C. 1997. Organizational design and management characteristics of new product development systems and (their interactions with) their context, related to performance. *Book*. p47.
- Dorst, K. and Cross, N. 2001. Creativity in the design process: co-evolution of problem–solution. *Design Studies*. **22**(5), pp.425-437.
- Dym, C.L., Agogino, A.M., Eris, O., Frey, D.D. and Leifer, L.J. 2005. Engineering Design Thinking, Teaching, and Learning. *Journal of Engineering Education*. **94**(1), pp.103-120.
- Eckert, C.M., Isaksson, O. and Earl, C.F. 2014. Design margins as a key to understanding design iteration. In: *ASME 2014 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*: American Society of Mechanical Engineers, pp.V007T007A022-V007T007A022.
- Eckert, C.M., Isaksson, O., Hallstedt, S., Malmqvist, J., Rönnbäck, A.Ö. and Panarotto, M. 2019. Industry trends to 2040. In: *Proceedings of the Design Society: International Conference on Engineering Design*: Cambridge University Press, pp.2121-2128.
- Eckert, C.M. and Stacey, M. 2010. What is a process model? Reflections on the epistemology of design process models. *Modelling and management of engineering processes*. Springer, pp.3-14.

- Edgeman, R., Hammond, S., Keller, C. and McGraw, J. 2020. Virtuous cycles: Organizational dynamics of innovation and excellence. *Total Quality Management & Business Excellence*. **31**(11-12), pp.1290-1306.
- Freeman, D.L. and Cameron, L.J.T.M.L.J. 2008. Research methodology on language development from a complex systems perspective. **92**(2), pp.200-213.
- Hoegl, M. and Weinkauff, K. 2005. Managing task interdependencies in Multi-Team projects: A longitudinal study. *Journal of Management Studies*. **42**(6), pp.1287-1308.
- Jankovic, M. and Eckert, C.M. 2016. Architecture decisions in different product classes for complex products. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*. **30**(3), pp.217-234.
- King, D.W. 1994. Communication by Engineers: A Literature Review of Engineers' Information Needs, Seeking Processes, and Use. In.
- Kline, S.J. and Rosenberg, N. 1986. An overview of innovation. The positive sum strategy: Harnessing technology for economic growth. *The National Academy of Science, USA*. **35**, p36.
- Le, H.N., Wynn, D.C. and Clarkson, P.J. 2010. Evaluating the positive and negative impact of iteration in engineering processes. In: Heisig, P., et al. eds. *Modelling and Management of Engineering Processes*. Springer-Verlag London Limited, pp.89-100.
- Love, P.E. 2002. Auditing the indirect consequences of rework in construction: a case based approach. *Managerial Auditing Journal*. **17**(3), pp.138-146.
- Maier, A.M., Hepperle, C., Kreimeyer, M., Eckert, C.M., Lindemann, U. and Clarkson, P.J. 2007. Associations between factors influencing engineering design communication. In: *DS 42: Proceedings of ICED 2007, the 16th International Conference on Engineering Design, Paris, France, 28.-31.07. 2007*, pp.649-650 (exec. Summ.), full paper no. DS642_P_164.
- Maier, J.F., Wynn, D.C., Biedermann, W., Lindemann, U. and Clarkson, P.J. 2014. Simulating progressive iteration, rework and change propagation to prioritise design tasks. *Research in Engineering Design*. **25**(4), pp.283-307.
- Marsh, J.R. 1997. *The capture and utilisation of experience in engineering design*. thesis, University of Cambridge.
- McCarthy, I.P., Tsinopoulos, C., Allen, P. and Rose-Anderssen, C. 2006. New product development as a complex adaptive system of decisions. *Journal of product innovation management*. **23**(5), pp.437-456.
- Meboldt, M., Matthiesen, S. and Lohmeyer, Q. 2013. The dilemma of managing iterations in time-to-market development processes. In: *Second International Workshop on the Modelling and Management of Engineering Processes (MMEP 2012)*: Eidgenössische Technische Hochschule Zürich.
- Meho, L.I. and Tibbo, H.R. 2003. Modeling the information-seeking behavior of social scientists: Ellis's study revisited. *Journal of the American society for Information Science and Technology*. **54**(6), pp.570-587.
- Abdel-Hamid, T. and Madnick, S.E. 1991. *Software project dynamics: an integrated approach*. Prentice-Hall, Inc.
- Adams, R. and J. Atman, C. 2000. *Characterizing Engineering Student Design Processes: An Illustration of Iteration*.

- Afacan, Y. and Demirkan, H. 2011. An ontology-based universal design knowledge support system. *Knowledge-based systems*. **24**(4), pp.530-541.
- Ahmadi, R. and Wang, R.H. 1999. Managing development risk in product design processes. *Operations Research*. **47**(2), pp.235-246.
- Ahmed, S., Wallace, K.M. and Blessing, L.T. 2003. Understanding the differences between how novice and experienced designers approach design tasks. *Research in Engineering Design*. **14**(1), pp.1-11.
- AitSahlia, F., Johnson, E. and Will, P.J.I.T.o.E.M. 1995. Is concurrent engineering always a sensible proposition? **42**(2), pp.166-170.
- Amigo, C.R., Iritani, D.R., Rozenfeld, H. and Ometto, A. 2013. Product Development Process Modeling: State of the Art and Classification. In: *Berlin, Heidelberg*. Springer Berlin Heidelberg, pp.169-179.
- Arp, R., Smith, B., Spear, A.D., Arp, R., Smith, B. and Spear, A.D. 2015. 85Introduction to Basic Formal Ontology I: Continuants. *Building Ontologies With Basic Formal Ontology*. The MIT Press, p.0.
- Artmann, C. 2009. Literature Review. In: Artmann, C. ed. *The Value of Information Updating in New Product Development*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp.1-31.
- Arundachawat, P., Roy, R., Al-Ashaab, A. and Shehab, E. 2009. Design rework prediction in concurrent design environment: current trends and future research directions. In: *Proceedings of the 19th CIRP Design Conference—Competitive Design*: Cranfield University Press.
- Asimow, M. 1962. *Introduction to design*. Englewood Cliffs, NJ, Prentice-Hall.
- Asimow, M.J.I., Englewood Cliffs, NJ. 1962. *Introduction to Design* Prentice-Hall.
- Åström, K.J. and Murray, R.M. 2021. *Feedback systems: an introduction for scientists and engineers*. Princeton university press.
- Atman, C.J., Chimka, J.R., Bursic, K.M. and Nachtmann, H.L.J.D.s. 1999. A comparison of freshman and senior engineering design processes. **20**(2), pp.131-152.
- Austin, S., Steele, J., Macmillan, S., Kirby, P. and Spence, R. 2001. Mapping the conceptual design activity of interdisciplinary teams. *Design Studies*. **22**(3), pp.211-232.
- Badke-Schaub, P. and Gehrlacher, A. 2003. Patterns of decisions in design: leaps, loops, cycles, sequences and meta-processes. In: *DS 31: Proceedings of ICED 03, the 14th International Conference on Engineering Design, Stockholm*.
- Ballard, G. 2000. Positive vs negative iteration in design. In: *Proceedings Eighth Annual Conference of the International Group for Lean Construction, IGLC-6, Brighton, UK*, pp.17-19.
- Bhuiyan, N., Gerwin, D. and Thomson, V. 2004. Simulation of the New Product Development Process for Performance Improvement. *Management Science*. **50**(12), pp.1690-1703.
- Blessing, L.T. 1995. Comparison of design models proposed in prescriptive literature.
- Bogus, S.M., Molenaar, K.R. and Diekmann, J.E. 2006. Strategies for overlapping dependent design activities. *Construction Management and Economics*. **24**(8), pp.829-837.

- Borshchev, A. 2013. Multi-method modeling. In: *Proceedings of the 2013 Winter Simulation Conference: Simulation: Making Decisions in a Complex World*, pp.4089-4100.
- Borshchev, A. and Filippov, A. 2004. From system dynamics and discrete event to practical agent based modeling: reasons, techniques, tools. In: *Proceedings of the 22nd international conference of the system dynamics society*: Citeseer.
- Boudouh, T., Anghel, D.-C. and Garro, O. 2006. Design Iterations in a Geographically Distributed Design Process. In: ElMaraghy, H.A. and ElMaraghy, W.H. eds. *Advances in Design*. London: Springer London, pp.377-385.
- Braha, D. and Bar-Yam, Y. 2007. The Statistical Mechanics of Complex Product Development: Empirical and Analytical Results. *MANAGEMENT SCIENCE*. **53**(7), pp.1127-1145.
- Braha, D. and Maimon, O. 1998. The measurement of a design structural and functional complexity. *A Mathematical Theory of Design: Foundations, Algorithms and Applications*. Springer, pp.241-277.
- Braha, D. and Maimon, O. 2013. *A mathematical theory of design: foundations, algorithms and applications*. Springer Science & Business Media.
- Brown, S.L. and Eisenhardt, K.M. 1995. Product Development - Past Research, Present Findings, and Future-Directions. *Academy of Management Review*. **20**(2), pp.343-378.
- Browning, T.R. 1998. *Modeling and analyzing cost, schedule, and performance in complex system product development*. thesis, Massachusetts Institute of Technology, Sloan School of Management, Technology and Policy Program.
- Browning, T.R. 2018. Building models of product development processes: An integrative approach to managing organizational knowledge. *Systems Engineering*. **21**(1), pp.70-87.
- Browning, T.R. and Eppinger, S.D. 2002. Modeling impacts of process architecture on cost and schedule risk in product development. *Ieee Transactions on Engineering Management*. **49**(4), pp.428-442.
- Browning, T.R., Fricke, E. and Negele, H. 2006. Key concepts in modeling product development processes. *Systems Engineering*. **9**(2), pp.104-128.
- Bucciarelli, L.L. 1994. *Designing engineers*. MIT press.
- Campbell, M.I., Cagan, J. and Kotovsky, K. 1999. A-design: an agent-based approach to conceptual design in a dynamic environment. *Research in Engineering Design*. **11**(3), pp.172-192.
- Cannondale. 2020. *Cannondale bad boy*. [Online]. [Accessed 2022]. Available from: <https://www.cannondale.com/en-gb/bikes/active/urban/bad-boy>
- Cash, P., Hicks, B. and Culley, S. 2015. Activity Theory as a means for multi-scale analysis of the engineering design process: A protocol study of design in practice. *Design Studies*. **38**, pp.1-32.
- Cash, P., Škec, S. and Storga, M. 2019. The dynamics of design: exploring heterogeneity in meso-scale team processes. *Design Studies*. **64**, pp.124-153.

- Chahal, K. 2010. *A generic framework for hybrid simulation in healthcare*. thesis, Brunel University School of Information Systems, Computing and Mathematics
- Chakrabarti, A. and Blessing, L. 2016a. *Anthology of Theories and Models of Design*. Springer.
- Chakrabarti, A. and Blessing, L. 2016b. A review of theories and models of design. **95**(4), pp.325-340.
- Chakravarty, A.K. 2001. Overlapping design and build cycles in product development. *European Journal of Operational Research*. **134**(2), pp.392-424.
- Challenger, M., Vanherpen, K., Denil, J. and Vangheluwe, H. 2020. FTG+ PM: Describing Engineering Processes in Multi-Paradigm Modelling. *Foundations of Multi-Paradigm Modelling for Cyber-Physical Systems*. Springer, pp.259-271.
- Chirumalla, K. 2017. Clarifying the feedback loop concept for innovation capability: A literature review. In: *ISPIM Innovation Symposium: The International Society for Professional Innovation Management (ISPIM)*, p.1.
- Chiva-Gomez, R. 2004. Repercussions of complex adaptive systems on product design management. *Technovation*. **24**(9), pp.707-711.
- Cho, S.H. and Eppinger, S.D. 2001. Product development process modeling using advanced simulation.
- Cho, S.H. and Eppinger, S.D. 2005. A simulation-based process model for managing complex design projects. *Ieee Transactions on Engineering Management*. **52**(3), pp.316-328.
- Chusilp, P. and Jin, Y. 2006. Impact of mental iteration on concept generation. *Journal of Mechanical Design*. **128**(1), pp.14-25.
- Clark, K.B., Chew, W.B., Fujimoto, T., Meyer, J. and Scherer, F.M. 1987. Product Development in the World Auto Industry. *Brookings Papers on Economic Activity*. **1987**(3), pp.729-781.
- Clarkson, P.J. and Eckert, C.M. 2010. *Design process improvement: a review of current practice*. Springer Science & Business Media.
- Clarkson, P.J. and Hamilton, J.R. 2000. 'Signposting', a parameter-driven task-based model of the design process. *Research in Engineering Design*. **12**(1), pp.18-38.
- Clausing, D. 1994. *Total Quality Development*. New York: ASME Press,.
- Clegg, C.W., Robinson, M.A., Davis, M.C., Bolton, L.E., Pieniasek, R.L. and McKay, A. 2017. Applying organizational psychology as a design science: A method for predicting malfunctions in socio-technical systems (PreMiSTS). *Design Science*. **3**.
- Cooper, K.G. 1980. Naval ship production: A claim settled and a framework built. *Interfaces*. **10**(6), pp.20-36.
- Cooper, R.G. and Sommer, A.F. 2018. Agile–Stage-Gate for Manufacturers: Changing the Way New Products Are Developed Integrating Agile project management methods into a Stage-Gate system offers both opportunities and challenges. *Research-Technology Management*. **61**(2), pp.17-26.
- Costa, R. 2004. *Productive iteration in student engineering design projects*. thesis, Montana State University-Bozeman, College of Engineering.
- Costa, R. and Sobek, I.D.K. 2003. Iteration in engineering design: inherent and unavoidable or product of choices made? In: *ASME 2003 International*

design engineering technical conferences and Computers and information in engineering conference: American Society of Mechanical Engineers, pp.669-674.

Cronholm, S. and Göbel, H. 2022. Action design research: integration of method support. *International Journal of Managing Projects in Business*. **15**(8), pp.19-47.

Cross, N. 2001. Design cognition: Results from protocol and other empirical studies of design activity. *Design knowing and learning: Cognition in design education*. pp.79-103.

Crowder, R.M., Robinson, M.A., Hughes, H.P. and Sim, Y.-W. 2012. The development of an agent-based modeling framework for simulating engineering team work. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*. **42**(6), pp.1425-1439.

Cusumano, M. and Selby, R.W. 1997. How Microsoft builds software. *Communications of the Acm*. **40**(6), pp.53-61.

Cusumano, M.A. 1997. How Microsoft makes large teams work like small teams. *Sloan Management Review*. **39**(1), pp.9-&.

Cutkosky, M.R., Engelmores, R.S., Fikes, R.E., Genesereth, M.R., Gruber, T.R., Mark, W.S., Tenenbaum, J.M. and Weber, J.C. 1993. PACT: An experiment in integrating concurrent engineering systems. *Computer*. **26**(1), pp.28-37.

De Bruijn, H. and Herder, P.M. 2009. System and actor perspectives on sociotechnical systems. *IEEE Transactions on systems, man, and cybernetics-part A: Systems and Humans*. **39**(5), pp.981-992.

de Weerd-Nederhof, P.C. 1997. Organizational design and management characteristics of new product development systems and (their interactions with) their context, related to performance. *Book*. p47.

de Weerd-Nederhof, P.C. 2001. Qualitative case study research. The case of a Ph.D. research project on organizing and managing new product development systems. *Management decision*. **39**(7), pp.513-538.

Dehkordi, F.M., Thompson, A. and Larsson, T. 2012. Impacts of project-overload on innovation inside organizations: agent-based modeling. *International Journal of Social, Behavioral, Educational, Economic, Business and Industrial Engineering*. **6**(11), pp.2808-2813.

Djanatljev, A. and German, R. 2013. Prospective healthcare decision-making by combined system dynamics, discrete-event and agent-based simulation. In: *2013 Winter Simulations Conference (WSC)*: IEEE, pp.270-281.

Djanatljev, A. and German, R. 2015. Towards a guide to domain-specific hybrid simulation. In: *2015 Winter Simulation Conference (WSC)*: IEEE, pp.1609-1620.

Dorst, K. and Cross, N. 2001. Creativity in the design process: co-evolution of problem–solution. *Design Studies*. **22**(5), pp.425-437.

Dresch, A., Lacerda, D.P. and Antunes, J.A.V. 2015. Design science research. *Design science research*. Springer, pp.67-102.

Dubiel, B. and Tsimhoni, O. 2005. Integrating agent based modeling into a discrete event simulation. In: *Proceedings of the Winter Simulation Conference, 2005.*: IEEE, p.9 pp.

- Dullen, S., Verma, D. and Blackburn, M. 2019. Review of Research into the Nature of Engineering and Development Rework: Need for a Systems Engineering Framework for Enabling Rapid Prototyping and Rapid Fielding. *Procedia Computer Science*. **153**, pp.118-125.
- Dybå, T. and Dingsøy, T. 2008. Empirical studies of agile software development: A systematic review. *Information and Software Technology*. **50**(9), pp.833-859.
- Dym, C.L., Agogino, A.M., Eris, O., Frey, D.D. and Leifer, L.J. 2005. Engineering Design Thinking, Teaching, and Learning. *Journal of Engineering Education*. **94**(1), pp.103-120.
- Eastman, R.M. 1980. Engineering information release prior to final design freeze. *IEEE transactions on Engineering Management*. (2), pp.37-42.
- Eckert, C.M., Clarkson, P.J. and Zanker, W. 2004. Change and customisation in complex engineering domains. *Research in engineering design*. **15**(1), pp.1-21.
- Eckert, C.M., Isaksson, O. and Earl, C.F. 2014. Design margins as a key to understanding design iteration. In: *ASME 2014 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*: American Society of Mechanical Engineers, pp.V007T007A022-V007T007A022.
- Eckert, C.M., Isaksson, O., Hallstedt, S., Malmqvist, J., Rönnbäck, A.Ö. and Panarotto, M. 2019. Industry trends to 2040. In: *Proceedings of the Design Society: International Conference on Engineering Design*: Cambridge University Press, pp.2121-2128.
- Eckert, C.M. and Stacey, M. 2010. What is a process model? Reflections on the epistemology of design process models. *Modelling and management of engineering processes*. Springer, pp.3-14.
- Edgeman, R., Hammond, S., Keller, C. and McGraw, J. 2020. Virtuous cycles: Organizational dynamics of innovation and excellence. *Total Quality Management & Business Excellence*. **31**(11-12), pp.1290-1306.
- Eppinger, S.D., Whitney, D.E., Smith, R.P. and Gebala, D.A. 1994. A Model-Based Method for Organizing Tasks in Product Development. *Research in Engineering Design-Theory Applications and Concurrent Engineering*. **6**(1), pp.1-13.
- Fairley, R.E. and Willshire, M.J. 2005. Iterative rework: the good, the bad, and the ugly. *Computer*. **38**(9), pp.34-41.
- Fernandes, J., Henriques, E., Silva, A. and Moss, M.A. 2014. A method for imprecision management in complex product development. *Research in Engineering Design*. **25**(4), pp.309-324.
- Ford, D.N. and Serman, J. 1997. Dynamic modeling of product development processes.
- Ford, D.N. and Serman, J.D. 2003. The Liar's Club: concealing rework in concurrent development. *Concurrent Engineering*. **11**(3), pp.211-219.
- Frillici, F., Rotini, F. and Fiorineschi, L. 2016. Re-design the design task through TRIZ tools. In: *DS 84: Proceedings of the DESIGN 2016 14th International Design Conference*, pp.201-210.
- Galbraith, J.R. 1974. Organization design: An information processing view. *Interfaces*. **4**(3), pp.28-36.

- Gericke, K. and Blessing, L. 2011. Comparisons of design methodologies and process models across domains: a literature review. In: *DS 68-1: Proceedings of the 18th International Conference on Engineering Design (ICED 11), Impacting Society through Engineering Design, Vol. 1: Design Processes, Lyngby/Copenhagen, Denmark, 15.-19.08. 2011.*
- Gero, J.S. 1990. Design prototypes: a knowledge representation schema for design. *AI magazine*. **11**(4), pp.26-26.
- Goh, Y.M. and Ali, M.J.A. 2016. A hybrid simulation approach for integrating safety behavior into construction planning: An earthmoving case study. *Accident Analysis & Prevention*. **93**, pp.310-318.
- Griffin, A. 2002. Product development cycle time for business-to-business products. *Industrial Marketing Management*. **31**(4), pp.291-304.
- Grimm, V., Railsback, S.F., Vincenot, C.E., Berger, U., Gallagher, C., DeAngelis, D.L., Edmonds, B., Ge, J., Giske, J. and Groeneveld, J. 2020. The ODD protocol for describing agent-based and other simulation models: A second update to improve clarity, replication, and structural realism. *Journal of Artificial Societies and Social Simulation*. **23**(2).
- Gruber, T.R. 1993. A translation approach to portable ontology specifications. *Knowledge acquisition*. **5**(2), pp.199-220.
- Guindon, R. 1990. Designing the design process: Exploiting opportunistic thoughts. *Human-Computer Interaction*. **5**(2), pp.305-344.
- Ha, A.Y. and Porteus, E.L. 1995. Optimal timing of reviews in concurrent design for manufacturability. *Management Science*. **41**(9), pp.1431-1447.
- Haller, M., Lu, W., Stehn, L. and Jansson, G. 2015. An indicator for superfluous iteration in offsite building design processes. *Architectural Engineering and Design Management*. **11**(5), pp.360-375.
- Hao, Q., Shen, W., Zhang, Z., Park, S.-W. and Lee, J.-K. 2006. Agent-based collaborative product design engineering: An industrial case study. *Computers in industry*. **57**(1), pp.26-38.
- Hardebolle, C. and Boulanger, F. 2009. Exploring Multi-Paradigm Modeling Techniques. *SIMULATION*. **85**(11-12), pp.688-708.
- Hassannezhad, M., Cantamessa, M., Montagna, F. and Clarkson, P.J.J.J.o.M.D. 2019. Managing Sociotechnical Complexity in Engineering Design Projects. **141**(8).
- Hatchuel, A. 2001. Towards Design Theory and expandable rationality: The unfinished program of Herbert Simon. *Journal of management and governance*. **5**(3-4), pp.260-273.
- Hoedemaker, G.M., Blackburn, J.D. and Van Wassenhove, L.N. 1999. Limits to Concurrency*. *Decision Sciences*. **30**(1), pp.1-18.
- Hoegl, M. and Weinkauff, K. 2005. Managing task interdependencies in Multi-Team projects: A longitudinal study. *Journal of Management Studies*. **42**(6), pp.1287-1308.
- Huberman, B.A. and Wilkinson, D.M. 2005. Performance variability and project dynamics. *Computational & Mathematical Organization Theory*. **11**(4), pp.307-332.
- Hughes, H.P., Clegg, C.W., Robinson, M.A. and Crowder, R.M. 2012. Agent-based modelling and simulation: The potential contribution to organizational psychology. *Journal of Occupational and Organizational Psychology*. **85**(3), pp.487-502.

- Hwang, B.-G., Thomas, S.R., Haas, C.T. and Caldas, C.H. 2009. Measuring the impact of rework on construction cost performance. *Journal of Construction Engineering and Management*. **135**(3), pp.187-198.
- Hybs, I. and Gero, J.S. 1992. An evolutionary process model of design. *Design Studies*. **13**(3), pp.273-290.
- Iansiti, M. and MacCormack, A. 1997. Developing products on Internet time. *Harvard business review*. **75**(5), pp.108-118.
- Isaksson, O., Keski-Seppälä, S. and Eppinger, S.D. 2000. Evaluation of design process alternatives using signal flow graphs. *Journal of Engineering Design*. **11**(3), pp.211-224.
- Jankovic, M. and Eckert, C.M. 2016. Architecture decisions in different product classes for complex products. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*. **30**(3), pp.217-234.
- Jin, Y. and Benami, O. 2010. Creative patterns and stimulation in conceptual design. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing: AI EDAM*. **24**(2), p191.
- Jin, Y. and Chusilp, P. 2006. Study of mental iteration in different design situations. *Design Studies*. **27**(1), pp.25-55.
- Jin, Y. and Levitt, R.E. 1996. The virtual design team: A computational model of project organizations. *Computational & Mathematical Organization Theory*. **2**(3), pp.171-195.
- Joglekar, N.R., Yassine, A.A., Eppinger, S.D. and Whitney, D.E. 2001. Performance of coupled product development activities with a deadline. *Management Science*. **47**(12), pp.1605-1620.
- Jun, H. and Suh, H. 2008. A Modeling Framework for Product Development Process Considering its Characteristics. *IEEE Transactions on Engineering Management*. **55**(1), pp.103-119.
- Keller, R., Eckert, C.M. and Clarkson, P.J. 2008. Determining component freeze order: a redesign cost perspective using simulated annealing. In: *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, pp.333-342.
- Kennedy, B.M., Sobek, D.K. and Kennedy, M.N. 2014. Reducing rework by applying set-based practices early in the systems engineering process. *Systems Engineering*. **17**(3), pp.278-296.
- Kim, M., Zimmermann, T. and Nagappan, N. 2014. An empirical study of refactoring challenges and benefits at microsoft. *IEEE Transactions on Software Engineering*. **40**(7), pp.633-649.
- King, D.W. 1994. Communication by Engineers: A Literature Review of Engineers' Information Needs, Seeking Processes, and Use. In:
- Kline, S.J. and Rosenberg, N. 1986. An overview of innovation. The positive sum strategy: Harnessing technology for economic growth. *The National Academy of Science, USA*. **35**, p36.
- Kolodner, J.L. and Wills, L.M. 1996. Powers of observation in creative design. *Design Studies*. **17**(4), pp.385-416.
- Kratzer, J., Leenders, R.T.A. and Van Engelen, J.M. 2010. The social network among engineering design teams and their creativity: A case study among teams in two product development programs. *International Journal of Project Management*. **28**(5), pp.428-436.

- Krehmer, H., Meerkamm, H. and Wartzack, S. 2009. The product's degree of maturity as a measurement for the efficiency of design iterations. In: *DS 58-3: Proceedings of ICED 09, the 17th International Conference on Engineering Design, Vol. 3, Design Organization and Management, Palo Alto, CA, USA, 24.-27.08. 2009*.
- Krishnan, V., Eppinger, S.D. and Whitney, D.E. 1995. Accelerating product development by the exchange of preliminary product design information.
- Krishnan, V., Eppinger, S.D. and Whitney, D.E. 1997. A model-based framework to overlap product development activities. *Management science*. **43**(4), pp.437-451.
- Krishnan, V., Eppinger, S.D. and Whitney, D.E. 1997. Simplifying iterations in cross-functional design decision making. *Journal of Mechanical Design*. **119**(4), pp.485-493.
- Krishnan, V. and Ulrich, K. 2001. Product development decisions: A review of the literature. *Management Science*. **47**(1), pp.1-21.
- Kruchten, P. 2002. Planning an Iterative Project. *The Rational Edge*.
- Kvale, S. 2012. *Doing interviews*. Sage.
- Lawson, B. 2006. *How designers think: The design process demystified*. Routledge.
- Le, H.N. 2013. *A transformation-based model integration framework to support iteration management in engineering design*. PhD thesis, University of Cambridge.
- Le, H.N., Wynn, D.C. and Clarkson, P.J. 2010. Evaluating the positive and negative impact of iteration in engineering processes. In: Heisig, P., et al. eds. *Modelling and Management of Engineering Processes*. Springer-Verlag London Limited, pp.89-100.
- Le, H.N., Wynn, D.C. and Clarkson, P.J. 2012. Impacts of concurrency, iteration, design review, and problem complexity on design project lead time and error generation. *Concurrent Engineering-Research and Applications*. **20**(1), pp.55-67.
- Lee, J. and Hong, Y.S. 2015. Design freeze sequencing using Bayesian network framework. *Industrial Management & Data Systems*.
- Lee, S. and Peña-Mora, F. 2007. Understanding and managing iterative error and change cycles in construction. *System Dynamics Review: The Journal of the System Dynamics Society*. **23**(1), pp.35-60.
- Leenders, R.T.A., Van Engelen, J.M. and Kratzer, J. 2003. Virtuality, communication, and new product team creativity: a social network perspective. *Journal of Engineering and technology management*. **20**(1-2), pp.69-92.
- Lévárdy, V. and Browning, T.R. 2009. An adaptive process model to support product development project management. *IEEE Transactions on Engineering Management*. **56**(4), pp.600-620.
- Levitt, R.E., Thomsen, J., Christiansen, T.R., Kunz, J.C., Jin, Y. and Nass, C. 1999. Simulating project work processes and organizations: Toward a micro-contingency theory of organizational design. *Management Science*. **45**(11), pp.1479-1495.
- Liker, J.K. and James, M.M. 2006. The Toyota Way in Services: The Case of Lean Product Development. *Academy of Management Perspectives*. **20**(2), pp.5-20.

- Lindemann, U. 2014. Models of design. *An anthology of theories and models of design*. Springer, pp.121-132.
- Liu, A. and Lu, S.C.-Y. 2014. Alternation of analysis and synthesis for concept generation. *CIRP Annals*. **63**(1), pp.177-180.
- Loch, C., Mihm, J. and Huchzermeier, A. 2003. Concurrent engineering and design oscillations in complex engineering projects. *Concurrent Engineering*. **11**(3), pp.187-199.
- Loch, C.H. and Terwiesch, C. 1998. Communication and uncertainty in concurrent engineering. *Management Science*. **44**(8), pp.1032-1048.
- Love, P.E. 2002. Auditing the indirect consequences of rework in construction: a case based approach. *Managerial Auditing Journal*. **17**(3), pp.138-146.
- Love, P.E. and Edwards, D.J. 2004a. Determinants of rework in building construction projects. *Engineering, Construction and Architectural Management*.
- Love, P.E. and Edwards, D.J. 2004b. Forensic project management: The underlying causes of rework in construction projects. *Civil Engineering and Environmental Systems*. **21**(3), pp.207-228.
- Love, P.E., Edwards, D.J., Watson, H. and Davis, P. 2010. Rework in civil infrastructure projects: Determination of cost predictors. *Journal of construction Engineering and Management*. **136**(3), pp.275-282.
- Love, P.E., Mandal, P. and Li, H. 1999. Determining the causal structure of rework influences in construction. *Construction Management & Economics*. **17**(4), pp.505-517.
- Love, P.E.D. and Li, H. 2000. Quantifying the causes and costs of rework in construction. *Construction Management and Economics*. **18**(4), pp.479-490.
- Lüftenegger, E. 2020. Using Action Design Research for Co-creating Service-Dominant Business Artifacts between Academia and Industry. In: *EMISA Forum: Vol. 40, No. 1: De Gruyter*.
- MacCormack, A., Verganti, R. and Iansiti, M. 2001. Developing products on "Internet time": The anatomy of a flexible development process. *Management science*. **47**(1), pp.133-150.
- Madhusudan, T. 2005. An agent-based approach for coordinating product design workflows. *Computers in Industry*. **56**(3), pp.235-259.
- Maher, M.L. 2000. A model of co-evolutionary design. *Engineering with Computers*. **16**(3-4), pp.195-208.
- Maier, A.M., Hepperle, C., Kreimeyer, M., Eckert, C.M., Lindemann, U. and Clarkson, P.J. 2007. Associations between factors influencing engineering design communication. In: *DS 42: Proceedings of ICED 2007, the 16th International Conference on Engineering Design, Paris, France, 28.-31.07. 2007*, pp.649-650 (exec. Summ.), full paper no. DS642_P_164.
- Maier, A.M. and Störrle, H. 2011. What are the characteristics of engineering design processes? In: *DS 68-1: Proceedings of the 18th International Conference on Engineering Design (ICED 11), Impacting Society through Engineering Design, Vol. 1: Design Processes, Lyngby/Copenhagen, Denmark, 15.-19.08. 2011*.
- Maier, J.F., Wynn, D.C., Biedermann, W., Lindemann, U. and Clarkson, P.J. 2014. Simulating progressive iteration, rework and change propagation to prioritise design tasks. *Research in Engineering Design*. **25**(4), pp.283-307.

- March, L. 1984. In *Developments in design Methodology. de The Logic of Design, London, (John Wiley. pp.265-276.*
- Marsh, J.R. 1997. *The capture and utilisation of experience in engineering design.* thesis, University of Cambridge.
- Martinec, T., Škec, S. and Štorga, M. 2017. Exploring the decomposition of team design activity. In: *DS 87-8 Proceedings of the 21st International Conference on Engineering Design (ICED 17) Vol 8: Human Behaviour in Design, Vancouver, Canada, 21-25.08. 2017, pp.229-238.*
- Martínez-Miranda, J., Aldea, A., Bañares-Alcántara, R. and Alvarado, M. 2006. TEAKS: Simulation of human performance at work to support team configuration. In: *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems: ACM, pp.114-116.*
- Masuch, M.J.A.S.Q. 1985. Vicious circles in organizations. pp.14-33.
- McCarthy, I.P., Tsinopoulos, C., Allen, P. and Rose-Anderssen, C. 2006. New product development as a complex adaptive system of decisions. *Journal of product innovation management. 23(5), pp.437-456.*
- McChrystal, G.S., Collins, T., Silverman, D. and Fussell, C. 2015. *Team of teams: New rules of engagement for a complex world.* Penguin.
- McComb, C., Cagan, J. and Kotovsky, K. 2015. Studying Human Design Teams via Computational Teams of Simulated Annealing Agents. In: *ASME 2015 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, V007T06A030.*
- McKay, A., Chittenden, R., Hazlehurst, T., de Pennington, A., Baker, R. and Waller, T. 2022. The derivation and visualization of supply network risk profiles from product architectures. *Systems Engineering. 25(5), pp.421-442.*
- McKay, A., Stiny, G.N. and de Pennington, A. 2016. Principles for the definition of design structures. *International journal of computer integrated manufacturing. 29(3), pp.237-250.*
- Meboldt, M., Matthiesen, S. and Lohmeyer, Q. 2013. The dilemma of managing iterations in time-to-market development processes. In: *Second International Workshop on the Modelling and Management of Engineering Processes (MMEP 2012): Eidgenössische Technische Hochschule Zürich.*
- Meho, L.I. and Tibbo, H.R. 2003. Modeling the information-seeking behavior of social scientists: Ellis's study revisited. *Journal of the American society for Information Science and Technology. 54(6), pp.570-587.*
- Mihm, J., Loch, C. and Huchzermeier, A. 2003. Problem-solving oscillations in complex engineering projects. *Management Science. 49(6), pp.733-750.*
- Mitchell, V.L. and Nault, B.R. 2007. Cooperative planning, uncertainty, and managerial control in concurrent design. *Management Science. 53(3), pp.375-389.*
- Moen, R.D. and Norman, C.L. 2010. Circling back. *Quality Progress. 43(11), p22.*
- Montagna, F. and Cantamessa, M. 2017. Who has the development process in his hands? *International Journal of Product Development. 22(3), pp.189-211.*
- Moore, D., Sauder, J. and Jin, Y. 2016. Exploring Dual-Processes of Iteration in Conceptual Design. *International Journal of Engineering Education. 32(3), pp.1385-1395.*

- Mykoniatis, K. and Angelopoulou, A. 2020. A modeling framework for the application of multi-paradigm simulation methods. **96**(1), pp.55-73.
- Nelson, R.G., Azaron, A. and Aref, S. 2016. The use of a GERT based method to model concurrent product development processes. *European Journal of Operational Research*. **250**(2), pp.566-578.
- Nikolic, I. and Ghorbani, A. 2011. A method for developing agent-based models of socio-technical systems. In: *2011 International Conference on Networking, Sensing and Control*: IEEE, pp.44-49.
- Nikolic, I. and Lukszo, Z. 2013. *Agent-based modelling of socio-technical systems*. Springer.
- Noy, N.F. and McGuinness, D.L. 2001. Ontology development 101: a guide to creating your first ontology. Stanford knowledge systems laboratory technical report KSL-01-05 and Stanford medical informatics technical report SMI-2001-0880. *Stanford*.
- Orlikowski, W.J. and Iacono, C.S. 2001. Desperately seeking the "IT" in IT research—a call to theorizing the IT artifact. *Information systems research*. **12**(2), pp.121-134.
- Osborne, S.M. 1993. *Product development cycle time characterization through modeling of process iteration*. thesis, Massachusetts Institute of Technology.
- Otte, J.N., Kiritsi, D., Ali, M.M., Yang, R., Zhang, B., Rudnicki, R., Rai, R. and Smith, B. 2019. An ontological approach to representing the product life cycle. *Applied Ontology*. **14**(2), pp.179-197.
- Pastor, O. 2016. Conceptual modeling of life: beyond the homo sapiens. In: *International Conference on Conceptual Modeling*: Springer, pp.18-31.
- Patrashkova-Volzdoska, R.R., McComb, S.A., Green, S.G. and Compton, W.D. 2003. Examining a curvilinear relationship between communication frequency and team performance in cross-functional project teams. *IEEE Transactions on Engineering Management*. **50**(3), pp.262-269.
- Perišić, M.M., Martinec, T., Štorga, M. and Kanduč, T. 2016. Agent-based simulation framework to support management of teams performing development activities. In: *DS 84: Proceedings of the DESIGN 2016 14th International Design Conference*, pp.1925-1936.
- Perišić, M.M., Štorga, M. and Podobnik, V. 2018. Agent-based modelling and simulation of product development teams. *Tehnički vjesnik*. **25**(Supplement 2), pp.524-532.
- Pessôa, M.V.P. and Trabasso, L.G. 2017. The product development system. *The Lean Product Design and Development Journey*. Springer, pp.3-18.
- Petersson, A.M. and Lundberg, J. 2016. Applying action design research (ADR) to develop concept generation and selection methods. *Procedia Cirp*. **50**, pp.222-227.
- Piccolo, S.A., Maier, A.M., Lehmann, S. and McMahon, C.A. 2019. Iterations as the result of social and technical factors: empirical evidence from a large-scale design project. *Research in Engineering Design*. **30**(2), pp.251-270.
- Pritsker, A.A.B. 1966. GERT: Graphical Evaluation and Review Technique, Part II. Probabilistic and Industrial Engineering. *J. Ind. Eng.* (6), pp.293-301.
- Rahmandad, H. and Hu, K. 2010. Modeling the rework cycle: capturing multiple defects per task. *System Dynamics Review*. **26**(4), pp.291-315.

- Regenwetter, L., Curry, B. and Ahmed, F. 2021. BIKED: A Dataset and Machine Learning Benchmarks for Data-Driven Bicycle Design. *arXiv e-prints*. parXiv: 2103.05844.
- Repenning, N.P. 2001. Understanding fire fighting in new product development*. *Journal of Product Innovation Management: AN INTERNATIONAL PUBLICATION OF THE PRODUCT DEVELOPMENT & MANAGEMENT ASSOCIATION*. **18**(5), pp.285-300.
- Richardson, G.P. and Pugh, A.L. 1981. *Introduction to system dynamics modeling with DYNAMO*. MIT press Cambridge, MA.
- Ritchie, E. 1972. Planning and control of R & D activities. *Journal of the Operational Research Society*. **23**(4), pp.477-490.
- Robinson, M.A. 2010. An empirical analysis of engineers' information behaviors. *Journal of the American Society for information Science and technology*. **61**(4), pp.640-658.
- Robinson, M.A. 2016. Quantitative research principles and methods for human-focused research in engineering design. *Experimental Design Research*. Springer, pp.41-64.
- Roemer, T.A. and Ahmadi, R. 2004. Concurrent crashing and overlapping in product development. *Operations research*. **52**(4), pp.606-622.
- Roemer, T.A., Ahmadi, R. and Wang, R.H. 2000. Time-cost trade-offs in overlapped product development. *Operations Research*. **48**(6), pp.858-865.
- Rycroft, R.W. and Kash, D.E. 1999. *The complexity challenge: Technological innovation for the 21st century*. Burns & Oates.
- Safoutin, M.J. 2003. *A methodology for empirical measurement of iteration in engineering design processes*. thesis, University of Washington Seattle.
- Safoutin, M.J. and Smith, R.P. 1996. The iterative component of design. In: *IEMC 96 Proceedings. International Conference on Engineering and Technology Management. Managing Virtual Enterprises: A Convergence of Communications, Computing, and Energy Technologies, 18-20 Aug 1996*, pp.564-569.
- Sargent, R., G. 2020. Verification and validation of simulation models: an advanced tutorial. In: *2020 Winter Simulation Conference (WSC): IEEE*, pp.16-29.
- Scheidegger, A.P.G., Pereira, T.F., de Oliveira, M.L.M., Banerjee, A. and Montevechi, J.A.B. 2018. An introductory guide for hybrid simulation modelers on the primary simulation methods in industrial engineering identified through a systematic review of the literature. *Computers & Industrial Engineering*. **124**, pp.474-492.
- Schlick, C.M., Duckwitz, S. and Schneider, S. 2013. Project dynamics and emergent complexity. *Computational and Mathematical Organization Theory*. **19**(4), pp.480-515.
- Schon, D.A. and Wiggins, G. 1992. Kinds of seeing and their functions in designing. *Design studies*. **13**(2), pp.135-156.
- Schrader, S., Riggs, W.M. and Smith, R.P. 1993. Choice over uncertainty and ambiguity in technical problem solving. *Journal of Engineering and Technology Management*. **10**(1-2), pp.73-99.
- Sein, M.K., Henfridsson, O., Purao, S., Rossi, M. and Lindgren, R. 2011. Action design research. *MIS quarterly*. pp.37-56.

- Sharman, D.M. and Yassine, A.A. 2004. Characterizing complex product architectures. *Systems Engineering*. **7**(1), pp.35-60.
- Shepherd, C. and Ahmed, P.K. 2000. NPD frameworks: a holistic examination. *European Journal of Innovation Management*.
- Shin, J.-H., Jun, H.-B., Kiritsis, D. and Xirouchakis, P. 2011. A decision support method for product conceptual design considering product lifecycle factors and resource constraints. *The International Journal of Advanced Manufacturing Technology*. **52**(9), pp.865-886.
- Singh, H., Cascini, G., Casakin, H. and Singh, V. 2019. A computational framework for exploring the socio-cognitive features of teams and their influence on design outcomes. In: *Proceedings of the Design Society: International Conference on Engineering Design*: Cambridge University Press, pp.1-10.
- Singh, V., Dong, A. and Gero, J.S. 2013. Social learning in design teams: The importance of direct and indirect communications. *AI EDAM*. **27**(2), pp.167-182.
- Škec, S., Cash, P. and Štorga, M. 2017. A dynamic approach to real-time performance measurement in design projects. *Journal of Engineering Design*. **28**(4), pp.255-286.
- Smith, R., P. and Eppinger, S., D. 1993. *Characteristics and models of iteration in engineering design*. International Motor Vehicle Program, Massachusetts Institute of Technology.
- Smith, R., P. and Eppinger, S.D. 1997a. Identifying Controlling Features of Engineering Design Iteration. *Management Science*. **43**(3), pp.276-293.
- Smith, R., P. and Eppinger, S.D. 1997b. A predictive model of sequential iteration in engineering design. *Management Science*. **43**(8), pp.1104-1120.
- Smith, R., P., Eppinger, S.D. and Gopal, A. 1992. Testing an engineering design iteration model in an experimental setting.
- Smith, R., P. and Morrow, J., A. 1999. Product development process modeling. *Design Studies*. **20**(3), pp.237-261.
- Smith, R., P. and Tjandra, P. 1998. Experimental observation of iteration in engineering design. *Research in Engineering Design*. **10**(2), pp.107-117.
- Smith, R.P. and Leong, A. 1998. An observational study of design team process: A comparison of student and professional engineers.
- Sosa, M.E., Eppinger, S.D. and Rowles, C.M. 2004. The misalignment of product architecture and organizational structure in complex product development. *Management science*. **50**(12), pp.1674-1689.
- Sulmall. 2022. *Bycycle design with BikeCad*. [Online]. [Accessed 2022]. Available from: https://www.sulmall.com/?product_id=239153977_35
- Szejka, A.L., Canciglieri Jr, O., Panetto, H., Rocha Loures, E. and Aubry, A. 2017. Semantic interoperability for an integrated product development process: a systematic literature review. *International Journal of Production Research*. **55**(22), pp.6691-6709.
- Tapia, F., McKay, A. and Robinson, M. 2021. Simulation of Feedback Loops in Engineering Design. *Proceedings of the Design Society*. **1**(Proceedings of the Design Society), pp.2661-2670.
- Taylor, T. and Ford, D.N. 2006a. Tipping point failure and robustness in single development projects. *System Dynamics Review*. **22**(1), pp.51-71.

- Taylor, T. and Ford, D.N. 2006b. Tipping point failure and robustness in single development projects. *System Dynamics Review: The Journal of the System Dynamics Society*. **22**(1), pp.51-71.
- Terwiesch, C. and Loch, C.H. 1999. Measuring the Effectiveness of Overlapping Development Activities. *Management Science*. **45**(4), p455.
- Terwiesch, C., Loch, C.H. and Meyer, A.D. 2002. Exchanging preliminary information in concurrent engineering: Alternative coordination strategies. *Organization Science*. **13**(4), pp.402-419.
- Thomke, S. and Bell, D.E. 2001. Sequential testing in product development. *Management Science*. **47**(2), pp.308-323.
- Thomke, S.H. 1997. The role of flexibility in the development of new products: An empirical study. *Research Policy*. **26**(1), pp.105-119.
- Thomke, S.H. 1998. Managing Experimentation in the Design of New Products. *Management Science*. **44**(6), pp.743-762.
- Toye, G., Cutkosky, M.R., Leifer, L.J., Tenenbaum, J.M. and Glicksman, J. 1994. SHARE: A methodology and environment for collaborative product development. *International journal of intelligent and cooperative information systems*. **3**(02), pp.129-153.
- Tsoukas, H. and e Cunha, M.P. 2017. On Organizational Circularity. *The Oxford Handbook of Organizational Paradox*. p393.
- Tsvetovat, M. and Carley, K.M. 2004. Modeling complex socio-technical systems using multi-agent simulation methods. *KI*. **18**(2), pp.23-28.
- Ulrich, K. 1995. The role of product architecture in the manufacturing firm. *Research policy*. **24**(3), pp.419-440.
- Ulrich, K. and Eppinger, S.D. 2012. *Product design and development*. Fifth edition. ed. Maidenhead: McGraw-Hill.
- Wallace, K.M. and Ahmed, S. 2003. How engineering designers obtain information: Human behaviour in design. *Human Behaviour in design: Individuals, teams, tools*. Springer Verlag, pp.184-194.
- Wang, J.X., Tang, M.X., Song, L.N. and Jiang, S.Q. 2009. Design and implementation of an agent-based collaborative product design system. *Computers in industry*. **60**(7), pp.520-535.
- Wang, Z. and Yan, H.-S. 2005. Optimizing the concurrency for a group of design activities. *IEEE Transactions on Engineering Management*. **52**(1), pp.102-118.
- Ward, A., Liker, J.K., Cristiano, J.J. and Sobek, D.K. 1995. The second Toyota paradox: How delaying decisions can make better cars faster. *Sloan management review*. **36**, pp.43-43.
- Whitworth, B. 2009. The Social Requirements of Technical Systems.
- Wynn, D.C. 2007. *Model-based approaches to support process improvement in complex product development*. thesis, University of Cambridge.
- Wynn, D.C. and Clarkson, P.J. 2005. Models of designing. *Design process improvement*. Springer, pp.34-59.
- Wynn, D.C. and Clarkson, P.J. 2017. Process models in design and development. *Research in Engineering Design*. **29**(2), pp.161-202.
- Wynn, D.C. and Clarkson, P.J. 2021. Improving the engineering design process by simulating iteration impact with ASM2. 0. *Research in Engineering Design*. **32**(2), pp.127-156.

- Wynn, D.C. and Eckert, C.M. 2017. Perspectives on iteration in design and development. *Research in Engineering Design*. **28**(2), pp.153-184.
- Wynn, D.C., Eckert, C.M. and Clarkson, P.J. 2007. Modelling iteration in engineering design.
- Wynn, D.C., Eckert, C.M. and Clarkson, P.J. 2019. Research into the design and development process: some themes and an overview of the special issue. *Research in Engineering Design*. **30**(2), pp.157-160.
- Wynn, D.C. and Maier, A.M. 2022. Feedback systems in the design and development process. *Research in Engineering Design*. pp.1-34.
- Wynn, D.C., Wyatt, D.F., Nair, S. and Clarkson, P.J. 2010. An introduction to the Cambridge advanced modeller.
- Yang, Q., Lu, T., Yao, T. and Zhang, B. 2014. The impact of uncertainty and ambiguity related to iteration and overlapping on schedule of product development projects. *International Journal of Project Management*. **32**(5), pp.827-837.
- Yang, Q., Yao, T., Lu, T. and Zhang, B. 2013. An overlapping-based design structure matrix for measuring interaction strength and clustering analysis in product development project. *IEEE Transactions on Engineering Management*. **61**(1), pp.159-170.
- Yassine, A.A. 2018. A Three-Dimensional View of Complex Product Development Management. In: *2018 IEEE Technology and Engineering Management Conference (TEMSCON)*: IEEE, pp.1-6.
- Yassine, A.A. 2019. Managing the Development of Complex Product Systems: An Integrative Literature Review. *IEEE Transactions on Engineering Management*. pp.1-18.
- Yassine, A.A. and Braha, D. 2003. Complex concurrent engineering and the design structure matrix method. *Concurrent Engineering-Research and Applications*. **11**(3), pp.165-176.
- Yassine, A.A., Chelst, K.R. and Falkenburg, D.R. 1999. A decision analytic framework for evaluating concurrent engineering. *IEEE Transactions on Engineering Management*. **46**(2), pp.144-157.
- Yassine, A.A., Joglekar, N., Braha, D., Eppinger, S.D. and Whitney, D. 2003. Information hiding in product development: the design churn effect. *Research in Engineering Design*. **14**(3), pp.145-161.
- Yassine, A.A., Sreenivas, R.S. and Zhu, J. 2008. Managing the exchange of information in product development. *European Journal of Operational Research*. **184**(1), pp.311-326.
- Yin, C. and McKay, A. 2018. Introduction to Modeling and Simulation Techniques. In: *Proceedings of ISCIIA 2018 and ITCA 2018*: Leeds.
- Yin, R.K. 1994. Discovering the future of the case study. Method in evaluation research. *Evaluation practice*. **15**(3), pp.283-290.
- Zhang, X., Luo, L., Yang, Y., Li, Y., Schlick, C.M. and Grandt, M. 2009. A simulation approach for evaluation and improvement of organisational planning in collaborative product development projects. *International Journal of Production Research*. **47**(13), pp.3471-3501.
- Zhang, X., Zhang, S., Li, Y. and Schlick, C. 2012. Task scheduling behaviour in agent-based product development process simulation. *International Journal of Computer Integrated Manufacturing*. **25**(10), pp.914-923.