

UNIVERSIDADE DE LISBOA  
INSTITUTO SUPERIOR TÉCNICO

**Requirements Change in Complex Product Development:  
Understanding Causes, Managing Uncertainty and  
Planning for Change**

**JOÃO MIGUEL VENTURA FERNANDES**

Supervisor: Doctor Elsa Maria Pires Henriques

Co-supervisor: Doctor Arlindo José de Pinho Figueiredo e Silva

Thesis approved in public session to obtain the PhD Degree in

**Leaders for Technical Industries**

Jury final classification: Pass with Distinction

**Jury**

**Chairperson:** Chairman of the IST Scientific Board

**Members of the Committee:**

Doctor Olivier Ladislav de Weck

Doctor Pedro Manuel Ponces Rodrigues de Castro Camanho

Doctor Elsa Maria Pires Henriques

Doctor Arlindo José de Pinho Figueiredo e Silva

Doctor César Figueiredo Pimentel

Doctor Michael Andrew Moss

**2015**



UNIVERSIDADE DE LISBOA  
INSTITUTO SUPERIOR TÉCNICO

**Requirements Change in Complex Product Development:  
Understanding Causes, Managing Uncertainty and  
Planning for Change**

**JOÃO MIGUEL VENTURA FERNANDES**

Supervisor: Doctor Elsa Maria Pires Henriques

Co-supervisor: Doctor Arlindo José de Pinho Figueiredo e Silva

Thesis approved in public session to obtain the PhD Degree in

**Leaders for Technical Industries**

Jury final classification: Pass with Distinction

**Jury**

**Chairperson:** Chairman of the IST Scientific Board

**Members of the Committee:**

Doctor Olivier Ladislav de Weck, Professor, Massachusetts Institute of Technology, USA

Doctor Pedro Manuel Ponces Rodrigues de Castro Camanho, Professor Catedrático da Faculdade de Engenharia, da Universidade do Porto

Doctor Elsa Maria Pires Henriques, Professora Associada do Instituto Superior Técnico, da Universidade de Lisboa

Doctor Arlindo José de Pinho Figueiredo e Silva, Professor Auxiliar do Instituto Superior Técnico, da Universidade de Lisboa

Doctor César Figueiredo Pimentel, Professor Auxiliar do Instituto Superior Técnico, da Universidade de Lisboa

Doctor Michael Andrew Moss, Corporate Specialist, Knowledge Management, Design Process Automation, Rolls-Royce plc

**Funding Institution**

Fundação para a Ciência e Tecnologia

**2015**

Requirements Change in Complex Product Development: Understanding Causes, Managing Uncertainty and Planning for Change

© JOÃO MIGUEL VENTURA FERNANDES, 2015

PhD Thesis  
Instituto Superior Técnico  
Universidade de Lisboa  
Avenida Rovisco Pais 1  
1049-001 Lisboa  
Portugal  
Telephone: (+351) 218 417 000

Printed at Instituto Superior Técnico  
Lisboa, February 2015

*Birds flying high  
Sun in the sky  
Breeze driftin' on by*

*It's a new dawn  
It's a new day  
It's a new life for me  
And I'm feeling good.*

ANTHONY NEWLEY



# Abstract

Requirements change is inevitable during product development and it is difficult to understand and manage for companies realizing large technical systems, such as cars, aircrafts or jet engines. This thesis tackles three aspects of requirements change in complex product development: understanding the causes of change, devising methods to predict and manage uncertainty and using models to investigate the effects of change and support planning.

The first part of this thesis reports findings from a large-scale empirical study which investigated the causes of change during the design process of a complex aerospace system developed by Rolls-Royce during the course of 6 years and examined a large dataset, containing over 1000 released changes.

The second part presents and evaluates a structured approach – the Imprecision Management Method (IMM) – supporting organizations understanding, quantifying and communicating the level of uncertainty in design information critical for complex development projects, such as uncertainty in requirements.

Lastly, the third part explores novel approaches to support planning and investigate the effects of change in project performance. To overcome limitations in current modelling approaches, this thesis describes the development of an Agent Model for Planning and rEsearch of eaRly dEsign (AMPERE) and presents initial results from simulation.

**Keywords:** requirements change, complex product development, requirements management, causes of requirements change, empirical study, uncertainty management, product development modelling, agent-based modelling, product development planning, planning for change

– This page is intentionally blank –



# Resumo

A modificação de requisitos é inevitável durante o desenvolvimento de produtos e difícil de compreender e gerir em empresas envolvidas na concepção de grandes sistemas técnicos, tais como carros, aviões ou motores a jacto. Esta tese aborda três aspectos da modificação de requisitos durante projectos de desenvolvimento complexos: a compreensão das causas, a formulação de métodos para gerir a incerteza e a utilização de modelos para investigar os seus efeitos e suportar o planeamento.

A primeira parte desta tese apresenta resultados de um estudo empírico às causas de alteração durante o processo de concepção de um sistema aeroespacial complexo desenvolvido pela Rolls-Royce durante seis anos, que examinou mais de 1000 alterações.

A segunda apresenta e avalia uma abordagem estruturada – o Método de Gestão de Imprecisão – destinada a suportar as organizações a compreender, quantificar e comunicar o nível de incerteza em informação crítica para os projectos, tal como a incerteza em requisitos.

Por fim, a terceira parte explora novas abordagens para suportar o planeamento e investigar os efeitos das alterações no desempenho dos projectos. De modo a ultrapassar as limitações de modelos existentes, esta tese descreve o desenvolvimento de um modelo baseado em agentes e apresenta resultados iniciais resultantes de simulação.

**Palavras-chave:** modificação de requisitos, projectos de desenvolvimento de produto complexos, gestão de requisitos, causas de modificação de requisitos, estudo empírico, gestão de incerteza, modelação de projectos de desenvolvimento de produto complexos, modelação baseada em agentes, planeamento de projectos de desenvolvimento de produto, planeamento da mudança

– This page is intentionally blank –

# List of Publications

This thesis is based on the work contained in four papers published in international scientific journals and conferences (papers I–IV). I am the first author of all the papers and the research published has been carried out by myself. I have discussed the scientific content of all papers with my supervisor, Doctor Elsa Henriques, and my co-supervisor, Doctor Arlindo Silva. Papers III and IV counted with additional contributions from Doctor Michael Moss from Rolls-Royce plc and went also through a review process in this company. Papers V and VI have been submitted with the latest research presented in this thesis and count with contributions from Prof. César Pimentel from the Intelligent Agents and Synthetic Characters Group of Instituto Superior Técnico. The complete list of publications resulting from this thesis is the following:

- I.** J. Fernandes, A. Silva, and E. Henriques. Modeling the impact of requirements change in the design of complex systems. In M. Aiguier, Y. Caseau, D. Krob, and A. Rauzy, editors, *Complex Systems Design & Management*, pages 151–164. Springer Berlin Heidelberg, 2013
- II.** J. Fernandes, E. Henriques, and A. Silva. A method for managing uncertainty levels in design variables during complex product development. In M. Aiguier, F. Boulanger, D. Krob, and C. Marchal, editors, *Complex Systems Design & Management*, pages 113–124. Springer International Publishing, 2014a
- III.** J. Fernandes, E. Henriques, A. Silva, and M. Moss. A method for imprecision management in complex product development. *Research in Engineering Design*, 25(4): 309–324, 2014b
- IV.** J. Fernandes, E. Henriques, A. Silva, and M. Moss. Requirements change in complex technical systems: an empirical study of root causes. *Research in Engineering Design*, 26(1):37–55, 2015
- V.** J. Fernandes, E. Henriques, A. Silva, and C. Pimentel. An agent-based approach to support planning for change during early design. Accepted to *ICED 2015*
- VI.** J. Fernandes, E. Henriques, A. Silva, and C. Pimentel. Modelling the dynamics of complex early design: an agent-based approach. Submitted

– This page is intentionally blank –

# Acknowledgements

This thesis is the outcome of my doctoral research and its realization counted with the support of many individuals and institutions which I am extremely grateful to.

First and foremost, I wish to express my sincere gratitude to my supervisor, Prof. Elsa Henriques, and to my co-supervisor, Prof. Arlindo Silva, for their continuous scientific support throughout the course of my research. Thank you for your valuable insights and comments and always taking the time to discuss ideas with open-mindedness and enthusiasm. I would also like to thank Prof. César Pimentel for participating in discussions and providing precious advices on the topic of agent-based simulations.

Secondly, I am grateful for all the support which has been provided by IST and the MIT-Portugal Program. I would like to recognize Prof. Manuel Freitas, Prof. António Ribeiro, Prof. Luis Faria and Prof. Mihail Fontul for providing inputs in various occasions. Many thanks also to Paula Taborda for support during stays abroad and for solving effectively many administrative issues. My acknowledgments go also to all the MIT-Portugal colleagues with whom I shared lunch or a coffee. A particular one goes to Jean-Loup Loyer, with whom I have shared many research discussions both at IST and at Rolls-Royce.

I would also like to acknowledge Rolls-Royce plc and many engineers who have supported my research. I would like to thank Michael Moss and Rob Farndon for gathering financial support, helping with internal procedures and contributing with their knowledge. To Kyle Martin, Carol McCorkindale and Heather Dunbar for sharing their experience during studies. To Andrew Wensley, Andrew White and Mark Stokes for opening the door of the Turbines Supply Chain Unit to my research. To all the engineers at Design Systems Engineering for the support during my stay. And to Carla Pepe, Jerome Berge, Pedro Cabrita, Sukhdev Chandarh, Joel Hora, Andreas Lewark, Albert Mari-Buye and Marc PonsPerez for helping out in Derby.

Furthermore, I wish to express my gratitude for the funding provided by “Fundação para a Ciência e Tecnologia” under the doctoral grant SFRH/BD/51107/ 2010 which allowed the research contributions contained in this thesis to be realized.

Lastly, this thesis is an accomplishment that would not have been possible without the encouragement provided by my family and friends. A special acknowledgment goes to my parents – Dionísio Fernandes and Maria Odete Ventura – and to Catarina Baptista: thank you for your unconditional support during this wonderful journey.

– This page is intentionally blank –

# Contents

<b>Abstract</b>	<b>v</b>
<b>Resumo</b>	<b>vii</b>
<b>List of Publications</b>	<b>ix</b>
<b>Acknowledgments</b>	<b>xi</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Fundamentals of complex system development . . . . .	2
1.1.1 The systems engineering approach . . . . .	3
1.1.2 The role of requirements . . . . .	5
1.2 Industrial needs . . . . .	6
1.3 Motivation . . . . .	10
1.4 Research questions . . . . .	12
1.4.1 Understanding the causes of change . . . . .	12
1.4.2 Predicting and managing uncertainty . . . . .	13
1.4.3 Planning for change . . . . .	14
1.5 Methodology . . . . .	15
1.5.1 Research paradigms and design research . . . . .	15
1.5.2 Doing design research . . . . .	17
1.5.3 Scope and methods . . . . .	18
1.6 Thesis outline . . . . .	21
 <b>I Understanding causes</b>	 <b>25</b>
 <b>2 The requirements management practice</b>	 <b>27</b>
2.1 The term “Requirement” . . . . .	29
2.2 Taxonomies . . . . .	31
2.3 The management process . . . . .	34
2.3.1 Process models . . . . .	34

2.3.2	Summary . . . . .	42
2.4	Support methods and tools . . . . .	44
2.4.1	Requirements elicitation . . . . .	44
2.4.2	Requirements prioritization . . . . .	47
2.4.3	Traceability analysis . . . . .	48
2.4.4	Change impact assessment . . . . .	51
2.5	Empirical contributions . . . . .	54
2.6	Review of progresses . . . . .	58
2.6.1	Opportunities for research . . . . .	58
2.6.2	Progress regarding research questions . . . . .	58
2.7	Summary . . . . .	59
<b>3</b>	<b>An empirical study of root causes of change</b>	<b>63</b>
3.1	Objectives . . . . .	64
3.2	Research setting . . . . .	65
3.2.1	Organizational context . . . . .	65
3.2.2	Company requirements management practices . . . . .	68
3.3	Research approach . . . . .	71
3.3.1	The case studied . . . . .	71
3.3.2	Reasons for selecting a case-study strategy . . . . .	72
3.3.3	Methodology and data sources . . . . .	73
3.3.4	Defining causes of change . . . . .	76
3.3.5	Sample design and representativeness . . . . .	78
3.4	Presentation and discussion of findings . . . . .	80
3.4.1	Evolution of active requirements and types of change . . . . .	80
3.4.2	Causes of change . . . . .	83
3.4.3	Inter-layer analysis . . . . .	90
3.4.4	Discussion . . . . .	93
3.5	Improvement guidelines for management . . . . .	95
3.5.1	Front-load the requirements engineering process . . . . .	96
3.5.2	Increase the level of concurrent requirements engineering . . . . .	96
3.5.3	Allocate more resources earlier . . . . .	98
3.5.4	Invest in practical training . . . . .	99
3.5.5	Set-up validation and verification systems . . . . .	99
3.5.6	Effects of the guidelines in the process . . . . .	100
3.6	Review of progresses . . . . .	101
3.7	Summary . . . . .	103



## **II Managing uncertainty 107**

### **4 A method for imprecision management 109**

4.1	Objectives . . . . .	110
4.2	Uncertainty in design . . . . .	112
4.2.1	Scope and an uncertainty definition . . . . .	114
4.2.2	Variability, aleatory and stochastic uncertainty . . . . .	115
4.2.3	Model uncertainty . . . . .	116
4.2.4	Design imprecision . . . . .	116
4.2.5	Research opportunities . . . . .	118
4.3	A method for imprecision management . . . . .	120
4.3.1	Collection of historical records . . . . .	120
4.3.2	Reconstruction of the variables' time evolution . . . . .	121
4.3.3	Statistical characterization of imprecision . . . . .	121
4.3.4	Communication of imprecision to new projects . . . . .	123
4.3.5	Knowledge update from new projects . . . . .	124
4.4	Empirical findings from an industrial case . . . . .	124
4.4.1	Setting . . . . .	125
4.4.2	Collecting historical records . . . . .	128
4.4.3	Time evolution of design variables . . . . .	129
4.4.4	Results from statistical analysis . . . . .	130
4.4.5	Communicating imprecision levels . . . . .	135
4.5	Discussion . . . . .	137
4.5.1	Strengths . . . . .	137
4.5.2	Limitations . . . . .	138
4.5.3	Integration with other change management approaches . . . . .	139
4.6	Review of progresses . . . . .	140
4.7	Summary . . . . .	143

## **III Planning for change 145**

### **5 Modelling complex and uncertain processes 147**

5.1	Purposes of process modelling . . . . .	148
5.2	Challenges for modelling . . . . .	150
5.2.1	Uncertainty . . . . .	150
5.2.2	Iteration . . . . .	151
5.2.3	Collaboration . . . . .	152
5.2.4	Adaptive behaviour . . . . .	154
5.3	Process modelling approaches . . . . .	155

5.3.1	Activity-based models . . . . .	156
5.3.2	Agent-based models . . . . .	165
5.3.3	System dynamics models . . . . .	168
5.3.4	Discussion . . . . .	169
5.4	Review of progresses . . . . .	173
5.5	Summary . . . . .	175
<b>6</b>	<b>An agent model of early design</b>	<b>177</b>
6.1	Objectives . . . . .	178
6.2	Research approach . . . . .	179
6.2.1	Approach selection . . . . .	179
6.2.2	Methodology . . . . .	181
6.3	Model development . . . . .	184
6.3.1	Vision . . . . .	184
6.3.2	Development platform selection . . . . .	184
6.3.3	Architectural design . . . . .	187
6.3.4	Agent definition . . . . .	191
6.3.5	Behaviours and interactions . . . . .	194
6.3.6	Effects of uncertainty and iteration . . . . .	197
6.3.7	Effects of collaboration and adaptation . . . . .	202
6.4	Exploratory simulations . . . . .	206
6.4.1	Simulation setup . . . . .	206
6.4.2	Process visualization . . . . .	210
6.4.3	Time evolution of internal variables . . . . .	213
6.4.4	Project performance evaluation . . . . .	216
6.4.5	Effects of change in performance . . . . .	218
6.4.6	Discussion . . . . .	220
6.5	Research progresses . . . . .	223
6.6	Summary . . . . .	224
<b>IV</b>	<b>Conclusions and Future Work</b>	<b>227</b>
<b>7</b>	<b>Conclusions</b>	<b>229</b>
7.1	Progresses made on research questions . . . . .	229
7.1.1	Understanding causes . . . . .	229
7.1.2	Managing uncertainty . . . . .	231
7.1.3	Planning for change . . . . .	233
7.2	Review of research contributions . . . . .	235
7.3	Opportunities for future research . . . . .	237

7.4 Conclusion . . . . .	239
<b>Bibliography</b>	<b>241</b>
<b>A Exploratory research</b>	<b>261</b>
<b>B Empirical study of root causes</b>	<b>265</b>
<b>C Agent model of early design</b>	<b>267</b>



# List of Figures

1.1	System life-cycle and typical activities and decision gates . . . . .	3
1.2	Systems engineering approaches used in the development of large technical systems . . . . .	5
1.3	The Rolls-Royce Trent XWB turbofan engine . . . . .	7
1.4	Sample of engineers and managers interviewed at Rolls-Royce during exploratory research. . . . .	8
1.5	Typical number of iterations before product release . . . . .	11
1.6	A methodology for implementing design research projects . . . . .	18
1.7	Scope of studies and methods used during the first part of research . . . .	19
1.8	Scope of studies and methods used during the second part of research . . .	20
1.9	Scope of studies and methods used during the third part of research . . . .	21
2.1	Overview of the literature review on requirements management . . . . .	29
2.2	The requirements management process model of Fiksel and Hayes-Roth . .	36
2.3	The spiral requirements engineering process model . . . . .	37
2.4	The requirements change management process model of Kotonya and Sommerville . . . . .	39
2.5	The requirements change management process of Pohl . . . . .	40
2.6	Overview of requirements and system design during space system development . . . . .	42
2.7	Thesis representation of the requirements management process . . . . .	43
2.8	The cost-value diagram for prioritizing requirements . . . . .	49
2.9	A simple requirements traceability matrix . . . . .	50
2.10	Overview of the Change Prediction Method . . . . .	53
3.1	Product introduction and life-cycle management process defined by Rolls-Royce . . . . .	67
3.2	General principles in requirements engineering at Rolls-Royce . . . . .	69
3.3	Documentation landscape at Rolls-Royce . . . . .	70
3.4	Time frame and milestones of the project investigated at Rolls-Royce . . .	72
3.5	Research approach and methods used in case-study . . . . .	75
3.6	Design principles to arrive to a representative stratified sample . . . . .	79

3.7	Evolution of system requirement throughout the course of the development of a new gas turbine . . . . .	81
3.8	Evolution of the types of change throughout the course of the project . . .	82
3.9	Mean proportion of each cause of change during phase dominated by modifications . . . . .	85
3.10	Mean proportion according to the type of change between PRD2 and PRD3	86
3.11	Mean proportion of each cause of change during phase dominated by additions	88
3.12	Mean proportion according to the type of change between PRD4 and PRD5	89
3.13	Proportion of change inside strata according to the type of requirement . .	89
3.14	Inter-layer analysis of the requirements engineering process . . . . .	91
3.15	Frequency of causes versus potential cost of change . . . . .	94
3.16	Representation of the effects of front-loading the requirements engineering process . . . . .	97
3.17	Simplified representation of the effects of improvement guidelines . . . . .	101
4.1	Uncertainties surrounding the value of a design variable . . . . .	118
4.2	The Imprecision Management Method . . . . .	122
4.3	High-level view of a gas turbine design process . . . . .	126
4.4	Time evolution of design variables during Project A . . . . .	130
4.5	Magnitude of Change found in the case-study . . . . .	132
4.6	Time evolution of the Magnitude of Change . . . . .	132
4.7	Distribution of the Time between Changes found in the case-study . . . . .	134
4.8	Time evolution of the Time between Changes . . . . .	135
4.9	Matrix of Imprecision found in the case-study . . . . .	136
5.1	Dimensions of design iteration . . . . .	153
5.2	Overview of the literature review on product development modelling . . . .	156
5.3	Activity-based models relying on precedence relationships . . . . .	158
5.4	Key features of activity-based adaptive models . . . . .	164
5.5	Key elements of agent-based models . . . . .	166
6.1	Vision guiding the development of the Agent Model for Planning and re-search of eaRly dEsign (AMPERE) . . . . .	185
6.2	Benchmark to agent development platforms . . . . .	186
6.3	Architectural design of the AMPERE . . . . .	189
6.4	Agent definition in AMPERE . . . . .	193
6.5	Effects of uncertainty and iteration . . . . .	200
6.6	Complete design process chart from a single simulation run . . . . .	212
6.7	Detailed view with annotations of design process chart . . . . .	213
6.8	Solution quality evolution during a single simulation run . . . . .	214

6.9	Risk perception evolution during a single simulation run . . . . .	214
6.10	Cost evolution during a single simulation run . . . . .	216
6.11	Histogram of solution quality . . . . .	217
6.12	Histogram of cumulative project cost . . . . .	218
6.13	Effects of change in project performance . . . . .	220
B.1	Template distributed during knowledge elicitation sessions . . . . .	266





# List of Tables

1.1	Research questions investigated in Thesis . . . . .	22
2.1	Methods supporting requirements elicitation . . . . .	45
2.2	Strengths and limitations of approaches supporting change impact assessment	55
3.1	Causes of change agreed with requirements managers . . . . .	77
3.2	Examples of changes in requirements . . . . .	86
5.1	Strengths and limitations of product development process modelling approaches . . . . .	174
6.1	Summary of desire selection according to agent beliefs . . . . .	195
6.2	Summary of communication messages . . . . .	197
6.3	Summary of setup for exploratory simulations . . . . .	208
A.1	Sample of engineers interviewed during exploratory research . . . . .	264
C.1	Description of internal elements in the Design Agent class. . . . .	267
C.2	Description of internal elements in the Task class. . . . .	270
C.3	Description of internal elements in the Solution class. . . . .	272



# Abbreviations

**ASM** Applied Signposting Model

**AMPERE** Agent Model for Planning and rEsearch of eaRly dEsign

**CAM** Cambridge Advanced Modeller

**CDR** Critical Design Review

**CPM** Critical Path Method

**CR** Concept Review

**CRD** Component Requirements Document

**BRD** Business Requirements Document

**DSM** Design Structure Matrix

**DMM** Domain Mapping Matrix

**FPE** Future Programs Engineering

**GERT** Graphical Evaluation and Review Technique

**IMM** Imprecision Management Method

**IPT** Integrated Program Team

**MoC** Magnitude of Change

**MoI** Matrix of Imprecision

**MoU** Memorandum of Understanding

**PERT** Program Evaluation and Review Technique

**PDR** Preliminary Design Review

**PRD** Product Requirements Document

**PRR** Production Readiness Review

**RFI** Request-for-Information

**RFP** Request-for-Proposal

**SCU** Supply Chain Unit

**SSRD** Sub-System Requirements Document

**SPADE** Smart Python multi-Agent Development Environment

**TbC** Time between Changes

**WED** Whole Enterprise Definition

**UML** Unified Modeling Language

# Chapter 1

## Introduction

Driven by the needs of social and economic welfare, many industrial firms thrive through the design and development of products that customers actually want to buy. Among those industrial firms, product development is truly challenging for organizations developing large technical systems. Large technical systems made by Man include cars, aircrafts, jet engines, satellites, oil refineries and big civil infrastructures like airports or power grids, for instance.

Such systems are constructs of interrelated parts forming a complex whole (Blanchard and Fabrycky, 2006) and functioning together to achieve goals which cannot be attained by the elements alone (NASA, 2007). The difficulty of designing and developing large technical systems arises from several fundamental reasons. The sheer size and scope of functions that must be performed by these systems forces firms to decompose them into many different pieces called sub-systems and components (Rechtin, 1991). Decomposition into a number of "building blocks" (Kossiakoff et al., 2011) is needed to manage complexity in development activities, production and assembly processes and operational and maintenance procedures. Moreover, each sub-system and component design solution needs to be developed concurrently of each other by multi-disciplinary engineering teams integrating many different specialities, such as mechanical engineering, electrical engineering, software engineering or materials science. Discipline specialists are required to engage into collaboration since sub-systems and component solutions normally require the use of advanced technologies delivering functionalities or performance characteristics

central to the behaviour of the overall system. In addition, there are also many interfaces and interdependencies – arising from decomposition – connecting different pieces of the system (Magee and de Weck, 2002) that need to be managed and resolved by engineering teams during component and sub-system design and integration activities to ensure that the system will function according to planned. Due to all of the above, delivering a large technical systems demands firms to plan, execute and manage a *complex product development process*.

## **1.1 Fundamentals of complex system development**

The realization of large technical systems normally follows a logical sequence based on the principle of phased development and on a gate-review process which controls and coordinates design efforts and provides baselines guiding the development, production and operation of the system (US DoD Systems Management College, 2001). The system's life-cycle is normally split into several stages, as shown in Figure 1.1.

The conceptual design stage is the first phase in complex system development and follows the approval of a new development initiative inside the organization. It typically includes the identification of stakeholder needs, requirements capture, concept exploration and design and the overall program planning. The preliminary design stage occurs subsequently to refine the system's conceptual solution and typically includes the preliminary definition of the major sub-systems and components. Detailed design and development follows with the full definition of sub-systems and components together with all the prototype development and testing activities and verification of internal and supplier manufacturing capabilities. Subsequent production approval triggers the start of large-scale manufacturing and the beginning of continuous system deliveries to the customer. The system is then put into operation and the last stage includes maintenance and logistic support to the customer, the release of product improvements and collection and analysis of operational data.

These and other typical activities performed during each stage of the system's life-cycle are also described in Figure 1.1, together with the typical decision gates comprised in each


CONCEPTUAL DESIGN	PRELIMINARY DESIGN	DETAILED DESIGN AND DEVELOPMENT	PRODUCTION	OPERATIONS AND SUPPORT		
<ul style="list-style-type: none"><li>• Need identification</li><li>• Requirements capture</li><li>• Concept design</li><li>• Selection of technical approach</li><li>• Functional definition of system</li><li>• Technology evaluation</li><li>• System/ program planning</li></ul>	<ul style="list-style-type: none"><li>• Functional analysis</li><li>• Trade-off studies</li><li>• Preliminary design</li><li>• Evaluation of design alternatives</li><li>• Acquisition plans</li><li>• Contracting</li><li>• Program implementation</li><li>• Major suppliers and supplier activities</li></ul>	<ul style="list-style-type: none"><li>• Sub-system / component detailed design</li><li>• Trade-off studies</li><li>• Prototype development</li><li>• Test and evaluation</li><li>• Verification of manufacturing processes</li><li>• Supplier activities</li><li>• Production planning</li></ul>	<ul style="list-style-type: none"><li>• Production ramp-up and large-scale production</li><li>• Supplier production activities and delivery</li><li>• System delivery to customer</li><li>• Acceptance testing</li><li>• Operational evaluation</li><li>• System assessment</li></ul>	<ul style="list-style-type: none"><li>• Operation in user environment</li><li>• Maintenance and logistic support to customers</li><li>• System modification for improvement</li><li>• Contractor support</li><li>• Field data collection and analysis</li></ul>		
TYPICAL DECISION GATES						
						
New Initiative Approval	Concept Approval	Preliminary Design Approval	Detailed Design Approval	Production Approval	Operational Approval	Deactivation Approval

Figure 1.1: System life-cycle and typical activities and decision gates in each stage. Adapted from Blanchard and Fabrycky (2006) and INCOSE (2006).

stage. Major technical and business reviews – e.g. Concept Review, Preliminary Design Review, Production Release Review – take place at each stage’s gate. Decision options at each gate normally encompass proceeding to the next stage, continuing or revisiting development activities of the current or preceding stages, putting project activities on hold or terminating the project (INCOSE, 2006).

### 1.1.1 The systems engineering approach

In addition, industrial organizations realizing large technical systems have increasingly adopted the systems engineering approach to product development during design and development activities. *Systems engineering* has been defined as the interdisciplinary framework concerned with the design, realization, operation and retirement of large technical systems (INCOSE, 2006). It emerged as an independent engineering discipline from the experience gained from military acquisition programs after the World War II (Sage and Armstrong, 2000) and it is an engineering management process focused on the optimization of the system as whole (US DoD Systems Management College, 2001). In face of opposing demands and conflicting constraints, systems engineering seeks solutions sat-

isfying the stakeholders' goals and maintaining a life-cycle perspective (NASA, 2007).

Being an engineering process, systems engineering is based on a hierarchical – often called "top-down" – comprehensive and iterative approach (US DoD Systems Management College, 2001). Fundamental principles of systems engineering include the capture and transformation of stakeholders' needs into requirements, the definition of a functional architecture and a compliant design solution and the flow-down of input information for development to the next level of the product's hierarchy. This functional decomposition and physical definition process starts normally at the highest system level and proceeds iteratively until component level is reached. The second set of basic principles of systems engineering comprises the subsequent and necessary "bottom-up" integration of components and sub-systems as well as validation and verification activities, which ensure that the whole system meets the original design intent. Verification also ensures that risks and opportunities are assessed and managed during development cycles.

These basic principles of the systems engineering' process are applied during all stages of complex product development but are particularly intense during conceptual design, preliminary design and detailed development. Various representations of the process appearing in systems engineering textbooks and publications have been captured by Kossiakoff et al. (2011) and Figure 1.2 reproduces the summary provided by this author. Figure 1.2 shows linear and "waterfall" type of processes containing various iterations that illustrate the necessity of revisiting previous activities to achieve coherence and compliance. It presents also the widespread systems engineering "V" process flow which provides both a life-cycle perspective and a view of the relationship between design activities and verification and validation activities. Finally, the spiral view of systems engineering is also illustrated in Figure 1.2, which explores the principle of continuously revisiting major engineering and management activities included in the process during all stages of development.



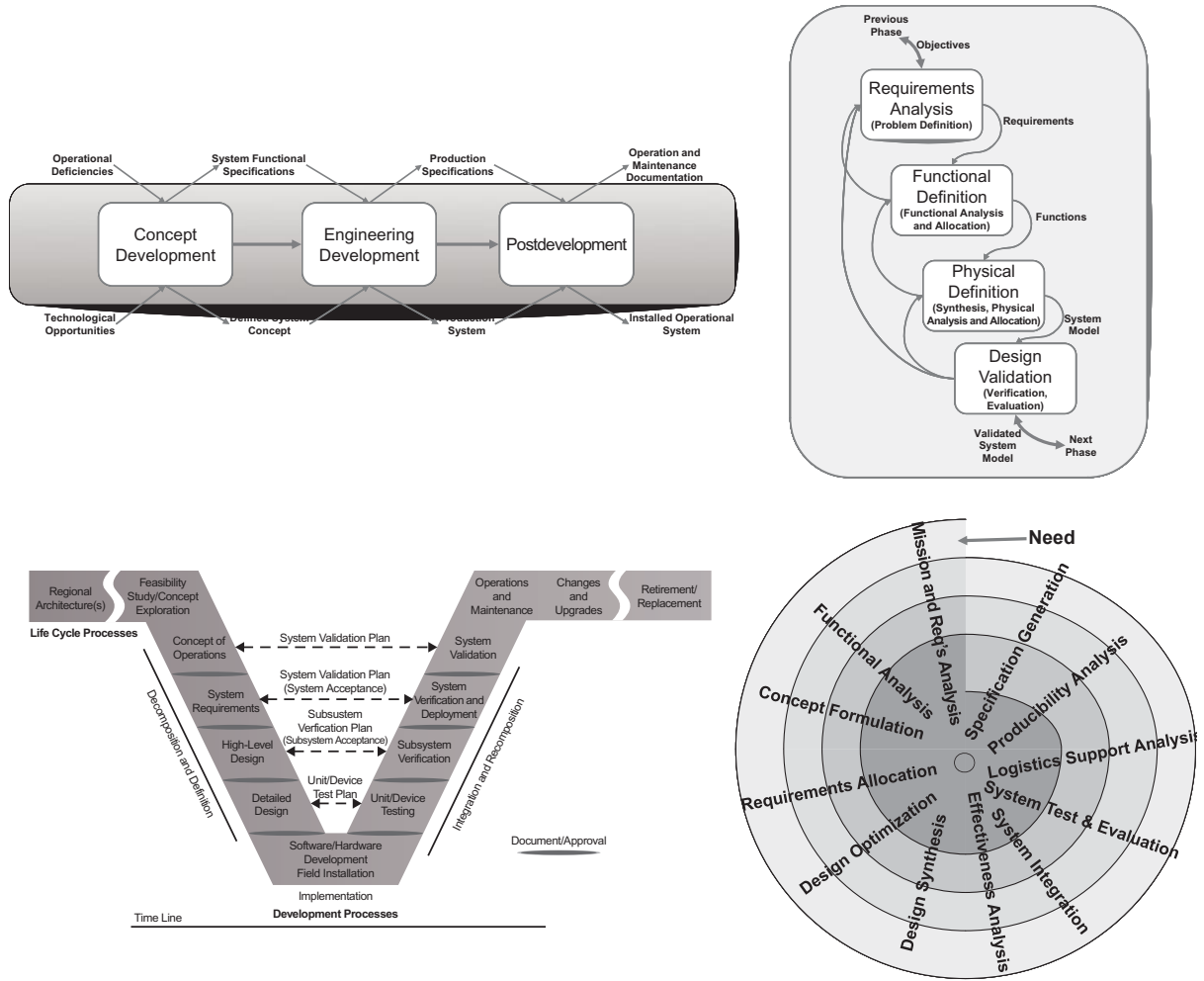


Figure 1.2: Systems engineering approaches used in the design and development of large technical systems [(Kossiakoff et al., 2011), p. 36].

### 1.1.2 The role of requirements

The processes represented in Figure 1.2 show also that requirements play a key role in the systems engineering approach to complex product development. At the beginning of any project, the needs of high-level stakeholders – customers, users, certification authorities and internal business functions, for instance – are captured and translated into system requirements. All needs are mapped from the language specific to each stakeholder into attributes and properties that can be used to guide engineers during development, production and operation of the system. Suh (1990) explains that this mapping process should create solution-independent requirements, which define the problem's domain. Thousands of requirements typically need to be captured by firms developing complex systems like

aircrafts, jet engines, satellites, oil refineries or big civil infrastructures.

Following this initial requirements capture and analysis, engineers then iterate to synthesize a functional architecture and a physical design solution for the system using engineering models, calculations and physical or virtual testing, until evidences show system requirements have been satisfied. The "waterfall" process represented in Figure 1.2 illustrates the previous sequence of activities. When evaluation and verification shows compliance, key system solution parameters are allocated as requirements – together with others flowing down directly from high-level stakeholders – to guide the development process of sub-systems. The process is then repeated for each lower level of the product's hierarchy and iterated until satisfactory solutions are found at all levels (NASA, 2007). Iterations are required at all levels since the engineers' understanding of both requirements and design solutions matures and changes over time. This process has been coined in engineering design as co-evolution (Maher et al., 1996; Dorst and Cross, 2001). The systems engineering approach is thus *requirements-driven* (INCOSE, 2006) and consequently requirements management is vital to complex product development.

## **1.2 Industrial needs**

The collaborative and iterative process of identifying all stakeholders, capturing, negotiating, flowing down, documenting and validating requirements is known as *requirements management* or *requirements engineering* (Kotonya and Sommerville, 1998; Robertson and Robertson, 2006; Pohl, 2010). Due to the amount of requirements arising during the development of large technical systems and the complexity of the design processes executed to synthesize these systems, requirements management processes are also intrinsically difficult.

In order to understand the main difficulties and needs of the engineers executing in practice such processes, the author conducted research at Rolls-Royce Plc, a world-leading provider of complex products such as jet engines for aerospace applications, gas turbines for energy and marine applications and nuclear power systems for military applications (Rolls-Royce, 2013). Rolls-Royce is a global player in the aerospace, defense, marine and

energy markets which employs over 40.000 people and has more than 59.000 gas turbines such as the one reproduced in Figure 1.3 in service around the world (Rolls-Royce, 2013).

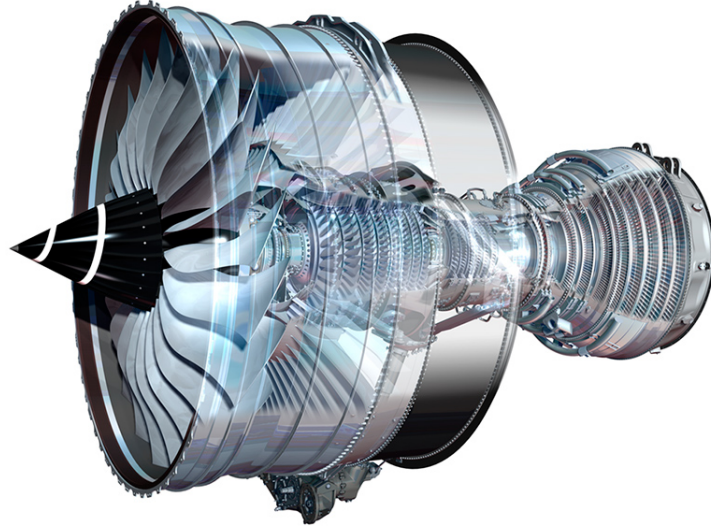


Figure 1.3: The Rolls-Royce Trent XWB turbofan engine (Rolls-Royce, 2013) designed to service in the new Airbus A350 XWB, a long-range two-engined wide-body aircraft.

A gas turbine is a *large technical system*, containing many sub-systems, hundreds of sub-assemblies and over two thousand different components (Whitney, 1988) that need to be perfectly designed and integrated to deliver numerous functionalities. Designing a gas turbine requires a *complex product development process*. It demands a large team of engineers with a wide range of technical expertise in aerodynamics, thermodynamics, mechanical design, material science and automation and control. These teams normally work distributed across multiple geographic locations, both within and outside the same country, and often in different time zones. Concurrent engineering takes place at system, sub-system and component levels from concept generation to product certification during more than 5 years. The development of gas turbines thus requires Rolls-Royce engineers to engage into a collaborative and distributed design process (Movahed-Khah et al., 2010) where a large number of design parameters and relationships needs to be determined. In addition, the product is designed to remain in service during more than 20 years, which makes its life-cycle unusually long.

The recognition that becoming effective at requirements management and systems engi-

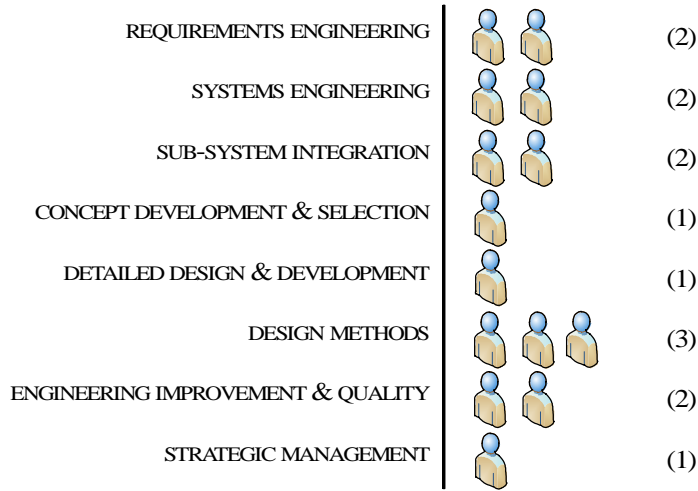


Figure 1.4: Sample of engineers and managers interviewed at Rolls-Royce during exploratory research.

neering is a source of competitive advantage in the market place has led this major manufacturer of gas turbines and other power systems to support this thesis. The author's research begun with an exploratory phase, where the needs of engineers and designers were investigated. A set of 14 semi-structured interviews was conducted to an heterogeneous sample of designers, engineers and managers which is shown in Figure 1.4. These interviews were based on an interview guide consisting of 30 pre-defined open questions organized around the requirements process, requirements uncertainty, response to uncertainty and risk, causes and impact of change and current challenges in managing change. Appendix A presents in detail all the questions included in the interview guide and the name, job title and main expertise of all interviewees.

The interviews followed the structure Introduction, Warm-up, Main Body, Cool-off and Closure recommended in Robson (2002) and lasted between 45 minutes and 1 hour with each interviewee. Sample selection was purposive but followed the method of maximum variation (Kuzel, 1999) to gain qualitative insight while minimizing sample selection bias through comparison of answers coming from people with very different expertise, experience and job responsibilities in the company. Despite of the pre-defined sequence of questions shown in Appendix A, considerable freedom was given to the interviewees and questions were often suppressed, added or re-sequenced, depending on the path taken by the interviewee.

One of the main findings from this exploratory research was that the classical paradigm of designing for "fixed" requirements has been abandoned in industrial practice. The designers, engineers and managers interviewed at Rolls-Royce – experienced individuals in the development of complex technical systems over long time scales – viewed requirements as *uncertain* information which evolves during the design and development process together with the environment they had been captured from. "Change is inevitable as the understanding and design matures", some mentioned. The causes of change were generally divided into externally or internally-driven changes. A wide range of causes was suggested, but there was no clear understanding among the interviewees about the relative frequency of each cause.

Another important finding from this initial research was that lack of knowledge about the level of uncertainty surrounding requirements and other design variables, particularly in those communicated by one design team as inputs to the activities of another team, creates many practical management issues. The author found that lack of quantitative knowledge characterizing uncertainty originated many questions in designers and engineers, such as:

- "Shall I wait for more accurate information about requirements before starting this activity?"
- "How much change should I expect at this stage of the development process?"
- "If I begin now with uncertain requirements what is the risk of having to do it all over again later on?"
- "Will I get any time saving if I begin working with uncertain inputs?"

These questions reflect the engineer's needs of comprehending and handling uncertainty and change in critical information – such as requirements – exchanged between design teams during complex collaborative and distributed engineering processes.

In addition, understanding and quantifying the effects of change in development projects was also found to be a key challenge. Changes in high-level requirements, changes arising during sub-system integration and late requirements change were considered to be the most "damaging" by the group interviewed, but the impact of change in projects was

thought to be “difficult to understand” and “usually underestimated”. The author found also that this difficulty of capturing the effects of change made particular project planning activities challenging, particularly during the early stages of product development, which is when uncertainties are known to be higher. For instance, specific challenges were captured from one of the specialists interviewed:

*“How can we plan resources taking into account the level of change expected in a new project? What is the right level of resources to deliver a solution with adequate quality during early design when requirements are changing often? How do we organize better the resources found to be needed?”*

Rolls-Royce engineers thus expressed their need for support estimating the effects of change in key project management variables – development time, cost or work quality, for instance – and for support in project planning particularly during early design (e.g. determining the adequate level of resources taking into account the expectable level of change). Exploratory research revealed also to the author the industrial receptiveness to evaluate novel approaches and tools to tackle these challenges, since it would provide additional decision-making support and that was considered to be highly valuable.

This thesis was thus born from these exploratory findings revealing the engineers’ needs of understanding the causes of requirements change, predicting and managing uncertainty and their need for support in planning for change during complex product development.

## **1.3 Motivation**

Prior academic research – Pikosz and Malmqvist (1998), Fricke et al. (2000) and Eckert et al. (2004b), for instance – has pointed out that requirements change is one of the main sources of engineering changes. Engineering change has been defined as an “alteration made to parts, drawings or software” that have been previously released during product development (Jarratt et al., 2011). A recent industrial survey performed by the Aberdeen Group (Brown, 2006) to 65 design and manufacturing companies to understand their product development practices showed that 40% of the companies experienced engineer-

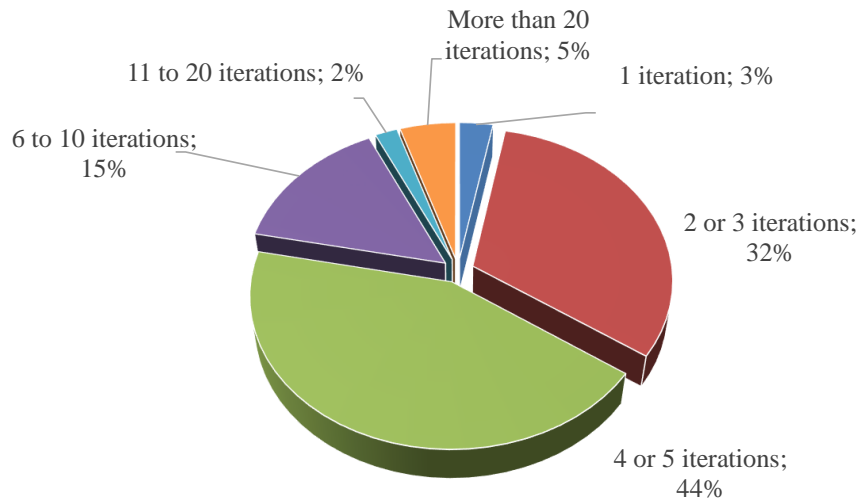


Figure 1.5: Typical number of design iterations before product release in design and manufacturing companies. Source: Brown (2006).

ing changes on a weekly basis and change caused additional rework and costs to 57% of the companies on a monthly or quarterly basis. Delays resulting from engineering change were experienced by 23% on a weekly basis, 26% on a monthly basis and by 24% of the companies on a quarterly basis.

The author's exploratory research also suggested that requirements change triggers changes to design solutions previously completed for the system, sub-system and components of complex products such as gas turbines. These changes precipitate new design iterations and the rework of many design activities and can propagate across sub-systems and components (Clarkson et al., 2004) due to design dependencies arising from decomposition, leading to higher development costs and time-to-market. The previous industrial survey showed that most companies normally go through 4 to 5 major design iterations before the product is released (Figure 1.5) and Cooper (1993) estimated the number of rework cycles within the aerospace industry to vary between 4 and 9.

Understanding and managing requirements uncertainty and change is important for product development-intensive industrial firms and *critical* for those executing long and complex development projects, due to this relationship between requirements change and unintentional design iterations and rework during projects. Rework can have a large impact in organizations such as Rolls-Royce, not only due to the financial resources needed

to revisit design activities but also from huge financial penalties that may be demanded from customers due to delays in the product's release. In addition, the difficulty of understanding when and how much change should be expected is illustrated in the day-to-day design process management questions highlighted in Section 1.2. The research motivation for this thesis is thus the **importance and difficulty of managing uncertainty and requirements change during complex product development projects**.

## 1.4 Research questions

Based on the industrial needs captured during exploratory research, the goal of this thesis is to answer three primary research questions and several secondary questions that derive from the three fundamental ones.

### 1.4.1 Understanding the causes of change

Managing requirements change is difficult because there is a wide range of potential uncertainties that affect the requirements engineering process. Because of that, the first primary research question targets comprehending how do requirements evolve and change during the course of complex product development projects:

**Q1** *How do requirements evolve and change during complex product development projects, such as the ones executed by Rolls-Royce?*

Following an initial investigation about what are the current best practices in requirements management, this thesis envisages to provide to academic research and industry an in-depth description of the underlying reasons that forces engineers to change requirements during projects, the evolution of the causes of change during different phases of the development process and the proportion of change arising from external and internal factors. The following secondary questions translate the goal of understanding in detail how requirements evolve and change:

**Q1-a** *What are the best practices in requirements management?*



**Q1-b** *What are the root causes of requirements change during the design of large technical systems?*

**Q1-c** *How do the causes of change vary during different phases of the product development process?*

**Q1-d** *How much change is externally and internally driven?*

## **1.4.2 Predicting and managing uncertainty**

The issue of finding pragmatic approaches to support designers and managers predicting and managing the level uncertainty that should be expected during complex development projects is the object of this thesis' second main research question:

**Q2** *How can the level of uncertainty in requirements be predicted and managed in industrial practice?*

The goal of predicting and managing uncertainty required, first and foremost, an understanding about the different types of uncertainty previously investigated in engineering design literature and about which are relevant to the definition of key design variables. Secondly, it was also envisaged that any practical approach developed by the author could demonstrate its industrial applicability, allowing both researchers and practitioners to understand and quantify how uncertainty evolved during real projects and what is the typical level of uncertainty that should be expected in new projects. Such issues originated a new group of secondary questions:

**Q2-a** *What are the main types of uncertainty in engineering design?*

**Q2-b** *How does the level of uncertainty in requirements and design variables varies during the course of complex product development projects?*

**Q2-c** *What is the typical level of uncertainty that should be expected by designers and engineers in new projects?*

### 1.4.3 Planning for change

Section 1.2 reported that engineers lacked support for planning taking into account expectable levels of change, particularly during early design stages which is when requirements are changing more often. During exploratory research, the author detected lack of models and tools allowing organizations to understand and quantify the effects of change in key project performance variables – such as development time, cost or solution quality – so that project planning activities can take them into account. This thesis argues that this is a major opportunity for research. Consequently, the third main research question is about comprehending:

**Q3** How can model-based approaches support planning for expectable levels of change during early stages of complex product development projects?

In order to support planning for change, it is necessary to investigate the relationship between changes in requirements and their effects in project performance. When considering possible research approaches to investigate cause-effect relationships of this kind, it must be realized that development projects of large technical systems occur seldom, spread over several years and project data is normally limited and problematic to collect. These research difficulties constrain the application of ethnographic or empirical data-driven approaches to investigate the effects of change in projects. An alternative research methodology to consider is however the use of *model-based simulation* since it provides a cost-effective way – through virtual experimentation – to represent a complex system and to understand and quantify the influence of key factors in its behaviour. This thesis thus explores models of product development to investigate the previous research question.

This third research question raises several additional questions. Firstly, it triggers the need to review previous model-based approaches to product development and to understand their relative strengths and limitations. Secondly, it drives the research of an appropriate modelling approach to capture the dynamics of early design, which is when requirements are changing more frequently and there is lack of planning support tools. Thirdly, it fosters the need to synthesize novel approaches that can support the estimation of global project performance during early stages taking into account the level of change that is

expected by designers and engineers. These issues set the last set of secondary research questions for this dissertation as:

**Q3-a** What approaches are available to model complex and uncertain design processes?

What are their main strengths and limitations?

**Q3-b** What approach is appropriate to capture the dynamics of early design, which is when requirements are changing often?

**Q3-c** How can model-based approaches support the estimation of project performance during early design stages taking into account the level of expectable change?

## 1.5 Methodology

This thesis is the outcome of what can be classified as a *design research project* executed by the author. This section introduces what is design research, what phases were implemented during the author's research project and what methodologies have been used to answer the different research questions presented in Section 1.4.

### 1.5.1 Research paradigms and design research

The purpose of research is to "explore, describe and explain" the world as we know it (Neuman, 2006) using structured approaches that ensure rigor and reliability in the results which arise from it. In his review of the different *research paradigms* used in Science, Neuman (2006) describes the different audiences and uses of research, time-dimensions and data collection techniques available:

- **Audience and use:** *basic research* is concerned with the derivation of fundamental knowledge and theories describing how and why a phenomenon in the world occurs and its main audience is the scientific community. Conversely, *applied research* is primarily occupied with finding results or solutions for specific problems of a community or organization. Its main audience is therefore the community or organization benefiting from the research.

- **Time-dimension:** *cross-sectional research* observes the world in a detailed manner at one point in time while *longitudinal research* examines normally one phenomenon at several points in time. On the other hand, *case-study research* is focused in achieving an in-depth understanding about a few selected features or phenomena over a particular duration of time.
- **Data-collection techniques:** *quantitative* techniques collect numeric data using, for instance, experiments, surveys or content analysis methods, while *qualitative* techniques use primarily words or pictures to transmit the findings from field, ethnographic or historical research methods.

Pahl and Beitz (2007) describe that it was not until the second half of the 20th Century that **Design** became a topic of scientific research with its own body of knowledge independent of the traditional – and much older – Engineering Sciences. The motivation for establishing a Science of Design arises from the socio-economic importance of product design and development, which greatly affects and shapes the way Humans live today. Recent decades have then seen numerous research contributions to the theoretical foundations of the Design Science. Some theoretical advances include for instance the Domain Theory (Andreasen, 1980), TRIZ (Altshuller, 1984), the Axiomatic Design Theory (Suh, 1990), the Theory of Synthesis (Takeda et al., 1999) or the CK-Theory (Hatchuel and Weil, 2003).

Discussions about the scope of design research and its boundaries connecting it to other scientific disciplines have been on-going within the research community. Eckert et al. (2003) argues that design research encompasses empirical studies of design behaviour and its evaluation, development of theoretical understanding, assessment of theory, development of tools and procedures and their evaluation and also introduction and assessment of such tools and procedures in industry. Basic and applied research across multiple time-dimensions are thus within the scope of design research.

Furthermore, the purpose of design research is ultimately to **understand design and improve the design practice** (Eckert et al., 2004a; Blessing and Chakrabarti, 2009). Blessing and Chakrabarti (2009) refer that "design is a complex multidisciplinary activity

involving people, knowledge, a product, an organization, processes, tools and methods”. Because of that, design research often needs to refer both to quantitative and qualitative research methodologies. The author’s point of view is that such diversity and interdisciplinarity makes design research a truly interesting and challenging scientific domain.

### 1.5.2 Doing design research

Concerned with the lack of structured methodologies to support researchers during design research projects, Blessing and Chakrabarti (2009) proposed also a design research methodology which relies on the execution of four main steps:

1. **Research Clarification** – an initial stage where the researcher establishes the overall research goals and the success criteria for evaluation of results arising from the project from a review of literature and/or exploratory research.
2. **Descriptive Study I** – a second stage where the researcher derives a detailed understanding of the current design situation and design problems that need to be solved using literature review and/or explanatory research methods.
3. **Prescriptive Study I** – a stage consisting on the elaboration of theories, methods, best practices or tools that support designers improving the current design situation and resolving design problems identified during the previous stage.
4. **Descriptive Study II** – a final stage where the impact of the support provided to designers is evaluated and its effectiveness in practice is measured.

Blessing and Chakrabarti refer that each of the method’s steps can be executed based on a **review-based study**, a **comprehensive study** or an **initial study** leading to seven different types of research projects. A review-based study is essentially supported in literature while a comprehensive study is an in-depth study including original results which have been produced by the researcher. An initial-study is one that involves a few first steps of a stage as a consequence of results from a previous stage. According to Blessing and Chakrabarti (2009), PhD projects should present *at least* one descriptive/prescriptive comprehensive study while long projects performed within a research

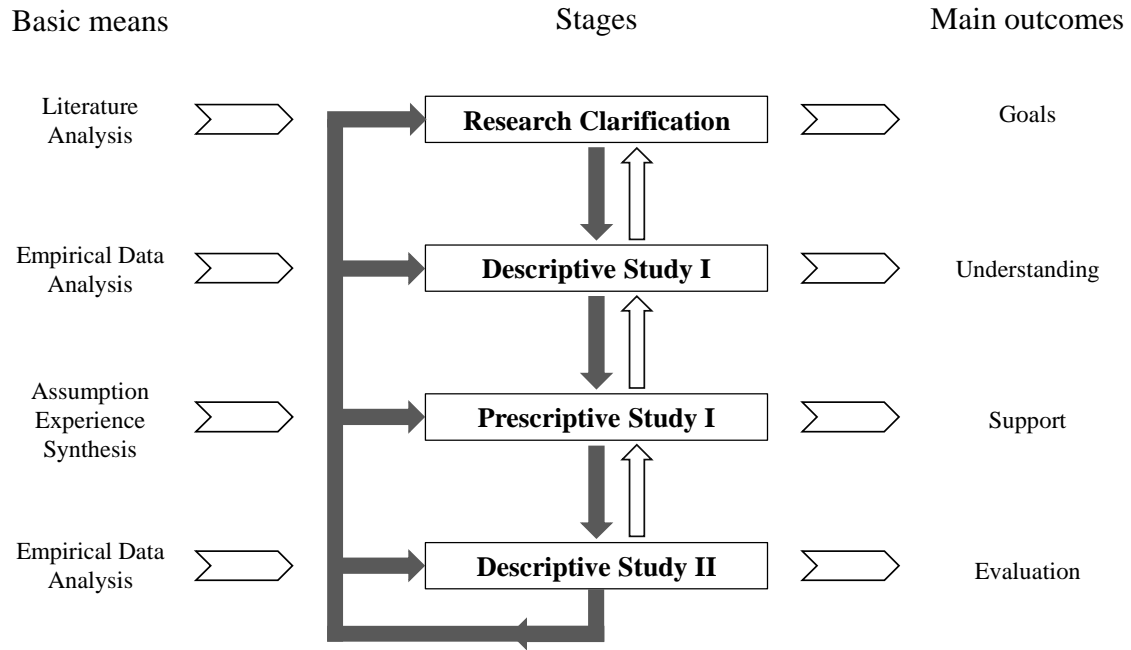


Figure 1.6: A methodology for implementing design research projects. Source: Blessing and Chakrabarti (2009).

group normally tackle all or several stages with such in-depth studies.

Although other approaches to Design Research have been proposed by Duffy and O'Donnell (1999) and Checkland (1981, 1999), the author selected Blessing and Chakrabarti's approach due to its higher generality. The framework represented in Figure 1.6 has thus guided the course of the author's research. The following section introduces the scope of the research performed by the author – mapping it against Blessing and Chakrabarti's design research methodology – and the methods used to answer each of the research questions presented in this thesis.

### 1.5.3 Scope and methods

After an exploratory research stage leading to the clarification of the research motivation and research questions, the first part of the author's work was to understand the existing practices at Rolls-Royce to manage requirements and to capture the root-causes of change during complex development projects. This was performed through a major case-study which involved an empirical collection of more than 1000 changes in requirements produced during a project (through archival research) and elicitation from experts of their

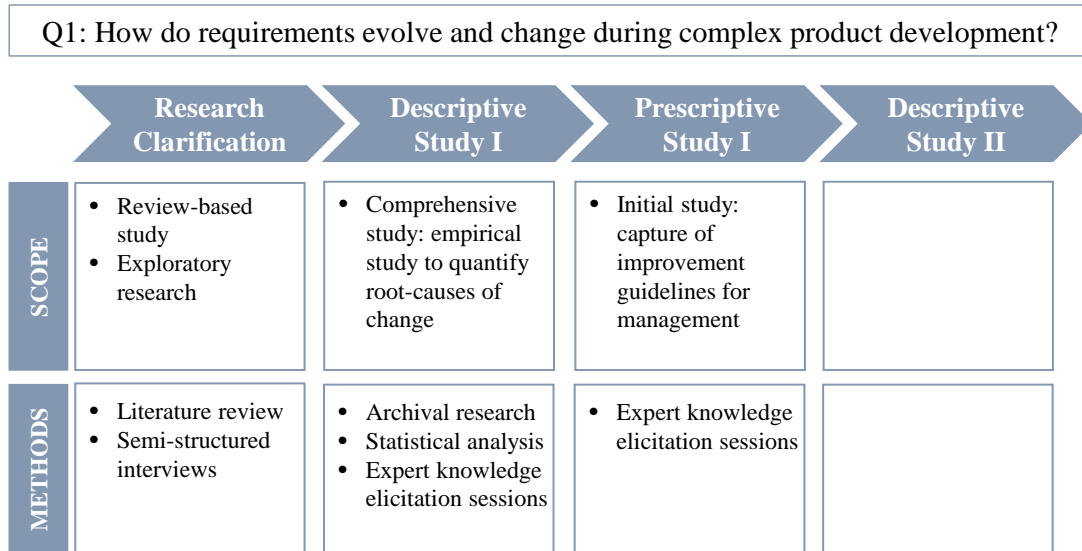


Figure 1.7: Scope of studies and methods used during the first part of the author’s research, following the design research methodology of Blessing and Chakrabarti (2009).

root-causes. Figure 1.7 shows that quantitative and qualitative methods were used allowing the author to characterize statistically the root-causes of change and its evolution during the project. Various improvement guidelines for management resulted also from the expert knowledge elicitation sessions. Following Blessing and Chakrabarti’s methodology, the scope of the research performed to answer the first group of questions thus encompassed both a comprehensive descriptive study and an initial prescriptive study (Figure 1.7).

The second part tackled how to predict and manage the level of uncertainty in requirements during complex development projects. As shown in Figure 1.8, the author begun by a review of prior literature about the topic of uncertainty in engineering design. Combined with the previous empirical research at Rolls-Royce, this led the author to the synthesis of a novel method aiming to support designers and engineers understanding, quantifying and communicating the typical level of uncertainty in requirements and other key design variables that should be expected during new projects (Figure 1.8). In addition, the method’s industrial applicability and potential was tested through a case-study. This study contributed to a greater understanding about how the level of uncertainty evolves during projects and provided a description of the typical level of uncertainty that can be expected by Rolls-Royce engineers in key requirements, namely those communicated

regularly between system and sub-system design teams. Figure 1.8 shows that this second phase comprised a first review-based descriptive study, a prescriptive study and an initial descriptive study according to the design research methodology followed in this thesis (Blessing and Chakrabarti, 2009).

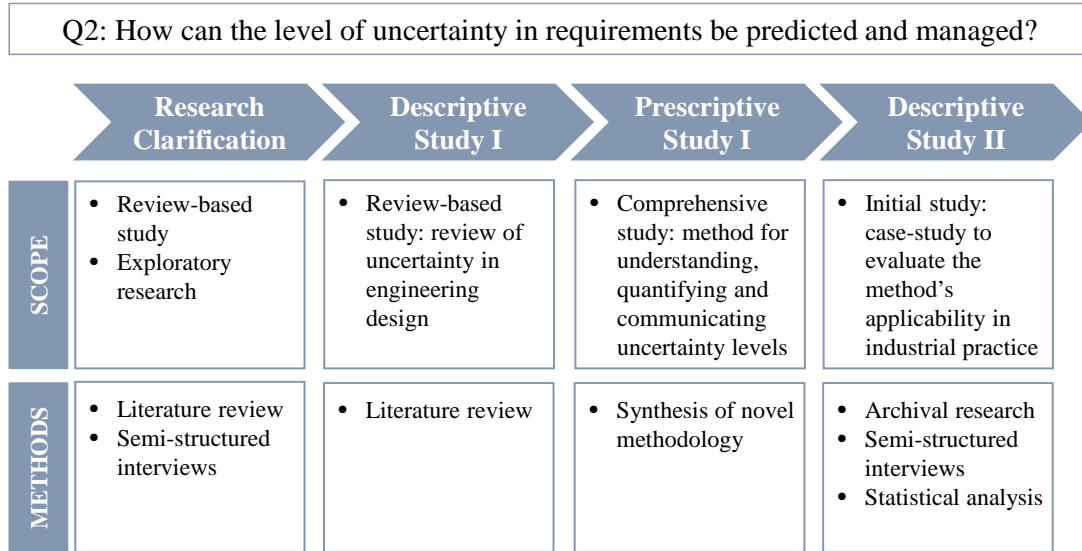


Figure 1.8: Scope of studies and methods used during the second part of the author's research, following the design research methodology of Blessing and Chakrabarti (2009).

Finally, the third part of the research aimed at supporting organizations planning complex product development processes during the early design stages taking into account expectable levels of requirements change. An in-depth literature review conducted the author to an evaluation of existing product development modelling approaches and allowed an assessment of the main strengths and limitations of the different methodologies and tools. Figure 1.9 depicts that this review led the author to the development of an agent-based model of early design stages and to an initial evaluation of its potential through exploratory simulations. Consequently, the research scope encompassed a review-based descriptive study, a prescriptive study with comprehensive original contributions and an initial descriptive evaluation of these contributions (Figure 1.9).



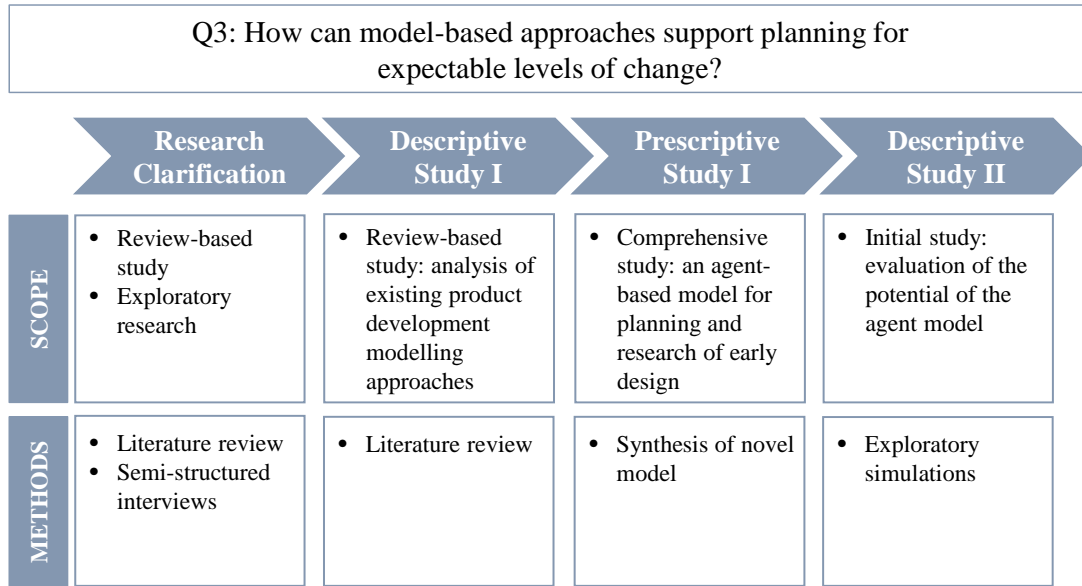


Figure 1.9: Scope of studies and methods used during the third part of the author’s research, following the design research methodology of Blessing and Chakrabarti (2009).

## 1.6 Thesis outline

The current Chapter provided the background – including findings from exploratory research – the research motivation and the key research questions that triggered this thesis. The remaining of the dissertation has been prepared to present, explore and discuss the studies summarized in Figures 1.7, 1.8 and 1.9 and the research findings arising from them. These studies were designed with the goal of providing answers to the questions defined in Section 1.4. In order to reflect the three groups of questions addressed and a discussion of contributions and conclusions from this research, the remaining of this thesis has been formally divided into **four parts** which embody seven main dissertation chapters. Table 1.1 presents which questions are investigated in each part and chapter. The dissertation’s introduction ends with an outline of each subsequent chapter:

### Part I Understanding causes

**Chapter 2. The requirements management practice** provides a critical review of the most relevant literature around requirements management and change management practices and discusses research gaps that guided the subsequent empirical study.

Table 1.1: Research questions investigated in each Part / Chapter.

Part / Chapter	Questions Investigated
Part I Understanding causes	Q1-all
Chapter 2 The requirements management practice	Q1-a
Chapter 3 An empirical study of root causes of change	Q1-b, Q1-c, Q1-d
Part II Managing uncertainty	Q2-all
Chapter 4 A method for imprecision management	Q2-a, Q2-b, Q2-c
Part III Planning for change	Q3-all
Chapter 5 Modelling complex and uncertain processes	Q3-a
Chapter 6 An agent model of early design	Q3-b, Q3-c
Part IV Conclusions and Future Work	All
Chapter 7 Conclusions	All

**Chapter 3. An empirical study of root causes of change** describes in detail the case-study performed by the author at Rolls-Royce to understand the root-causes of requirements change and its evolution during complex product development projects.

## Part II Managing uncertainty

**Chapter 4. A method for imprecision management** reviews prior literature about the main types of uncertainty in engineering design and introduces the method synthesized by the author to support designers and engineers understanding, quantifying and communicating the typical level of uncertainty in critical design variables such as requirements during complex product development projects. In addition, it describes empirical results from its application in an initial industrial evaluation case-study.

## Part III Planning for change

**Chapter 5. Modelling complex and uncertain processes** investigates available methods and tools to model complex product development processes and pro-

vides a critical discussion of the strengths and limitations of each approach available.

**Chapter 6. An agent model of early design** introduces an agent-based model of the early stages of complex product development aiming to support planning for change and presents initial simulations investigating the effects of change in project performance. In addition, it describes how key dynamics of early stages of product development such as uncertainty, iteration, collaboration and adaptive behaviours have been incorporated in the agent model.

## **Part IV Conclusions and Future Work**

**Chapter 7. Conclusions** summarizes the main findings and research contributions resulting from this dissertation and identifies and discusses several directions for future academic work according to the opportunities identified during the course of the author's research.



# Part I

## Understanding causes

The only source of knowledge is  
experience.

—ALBERT EINSTEIN



# Chapter 2

## The requirements management practice

As stated in Chapter 1, this thesis' first main research question targets understanding how requirements evolve and change during complex product development projects. Because of that, a review of past literature about the requirements management practice was necessary to place the research into the context of previous contributions.

The importance of engineering and managing requirements more effectively than others during development projects has captured the attention of various research and industrial communities during the last decades. In fact, requirements management appears as an important topic in various bodies of knowledge. It arises in engineering design literature which is concerned with the design and realization of physical systems – often comprising some kind of mechanical system – and has been traditionally focused in consumer products or capital goods (Pahl and Beitz, 2007). It is a central topic in software engineering, known to be primarily focused in the design, realization, operation and maintenance of software products such as information systems and embedded systems (Mills, 1980; Sommerville, 2010). And it is also a key issue in systems engineering literature which is concerned with the design, realization, management, operation and disposal of large technical systems (Blanchard and Fabrycky, 2006). Chapter 1 has previously referred that systems engineering addresses constructs composed of a large number of elements that are

normally needed to deliver functions, characteristics or properties only achievable when the elements are put together. These constructs may additionally exhibit properties arising from unanticipated interactions between the elements, known as emergent properties (Bertalanffy, 1968).

Since requirements management is relevant to engineering design, software engineering and systems engineering, the topic encompasses a *very large* number of publications. A review of requirements management literature performed by Jiao and Chen (2006) just within engineering design selected more than 100 publications. Conferences on requirements engineering within software produce roughly the same amount of new publications *every year*. Therefore, it is not feasible to provide an exhaustive review of all literature. Instead of that, this chapter provides a comprehensive review over the main requirements management sub-topics. The author presents key references to requirements management within engineering design, software and systems engineering to provide a rich discussion of literature containing multiple perspectives of this subject. And this chapter focuses on the publications which are relevant to the first research questions tackled by this dissertation.

Figure 2.1 illustrates the literature review on requirements management presented in this chapter, how it is organized across different sections and some of the key contributions analyzed in each sub-topic. Sections 2.1 and 2.2 begin with a survey of different requirements definitions and taxonomies proposed by various authors. Since requirements management is defined for industrial practice, Section 2.3 reviews process models found in engineering design, software and systems engineering literature, as shown in Figure 2.1. In addition, this thesis analyzes a wide range of methods and tools intending to support requirements management activities across Section 2.4. Methods used in requirements elicitation, prioritization, traceability analysis and change impact assessment are reviewed, discussed and compared. A review of key empirical research contributions on requirements management practice can be found in Section 2.5. The chapter ends with a discussion of research progresses and a summary of key aspects found in prior literature.



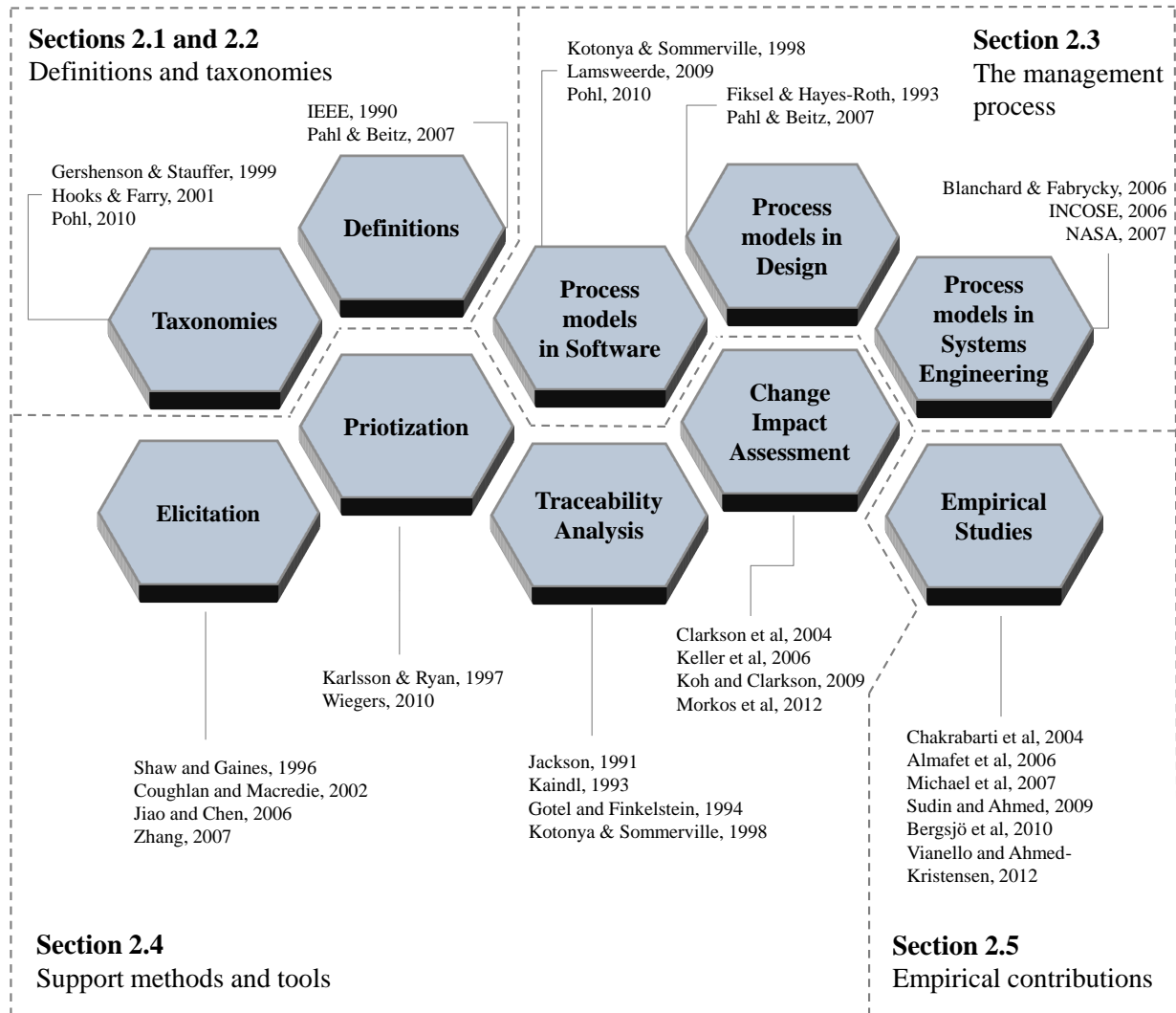


Figure 2.1: Overview of the literature review on requirements management performed in this chapter.

## 2.1 The term “Requirement”

Requirements have been defined within previous engineering design literature by Pahl and Beitz as:

*... attributes, properties or characteristics that the design solution must have in order to satisfy the identified customer needs (Pahl and Beitz, 2007)*

This is a general-purpose definition and normally an additional description of the *type* of

attribute, property or characteristic that the design solution must have is also incorporated. For instance, customer requirements usually refer to attributes expressed in the voice of the customer or the customers “wishes” (Griffin and Hauser, 1993). Mapping methods such as the House of Quality (Hauser and Clausing, 1988) allow then marketing and engineering personnel to translate customer requirements into functional or manufacturing requirements containing engineering terms associated respectively with the functional and manufacturing characteristics of the product to be realized.

Requirements are also found in design literature as *specifications*. Ulrich and Eppinger (2008) define specifications generically as “an unambiguous agreement of what the team will attempt to achieve in order to satisfy the identified customer needs”. Although this is a definition similar to the one from Pahl and Beitz (2007), specifications usually refer to target values imposed on engineering attributes. Because of that, specifications are typically only used in the product’s technical and engineering domain while requirements refer to a wider range of attributes and characteristics which are desired.

The term requirements is also extensively used in the context of software engineering. The Institute of Electrical and Electronics Engineers (IEEE), who defines standards specifically for software engineering, defines requirements in this context as:

*... a condition or capability that must be met or possessed by a system which is needed by users or demanded by standards, laws or internal stakeholders (IEEE, 1990)*

According to IEEE (1990) both documented as well as undocumented conditions or capabilities are called requirements. Moreover, requirements group the goals of users, the external constraints imposed by the environment or the needs and properties required inside the organization. Other authors like Kotonya and Sommerville (1998) have defined requirements as “statements of the services that the system should provide, specifications of the systems attributes and properties or constraints on the system”, which in practice also fits with the standard’s definition.

Taking into account the previous definitions found across multiple disciplines, the term requirement thus relates to qualities or constraints identified from multiple sources that the product must satisfy. Due to the wide scope of statements that fall under the requirements umbrella, different taxonomies have been developed and can be found in literature.

## 2.2 Taxonomies

The purpose of a requirements taxonomy is to help design teams organize and manage requirements during the design and development process and to foster the reuse of requirements information in future development projects (Gershenson and Stauffer, 1999a). Requirements have been classified according to different criteria and a review of the taxonomies found in literature is now presented.

In their famous and comprehensive book about engineering design, Pahl and Beitz (2007) classify requirements into demands and wishes according to the criteria of relative **importance** for the design process:

- *Demands* are defined as requirements that must be met in all circumstances, meaning that if any of these requirements is not fulfilled, the design solution is unacceptable.
- *Wishes* are important but not crucial requirements that should be considered whenever possible (Pahl and Beitz, 2007).

On the other hand, a requirements classification according to its **source** has been developed and presented by Gershenson and Stauffer (1999b), who named it as MOOSE Methodology of Organizing Specifications in Engineering. This taxonomy included four main types of requirements reflecting diverse sources of requirements captured during real projects:

- *End user requirements* refer to requirements captured from customer or end users through marketing research.

- *Corporate requirements* reflect the needs of corporate stakeholders and may consist of business requirements such as the production volume or the return on investment required for the project, for instance.
- *Regulatory requirements* arise from regulations and standards approved by governmental agencies regulating the industry where the firm is acting.
- *Technical requirements* consist of documented requirements created by the designers and engineers involved in the project based on its knowledge of the product.

In addition, Hooks and Farry (2001) considered the different aspects of the products **life-cycle** to propose an extensive requirements taxonomy which has been also classified by Dong (2002) as a subject-based classification. This taxonomy included for instance:

- *Functional requirements*
- *Performance requirements*
- *Physical requirements*
- *Aesthetic requirements*
- *Manufacturing requirements*
- *Assembly requirements*
- *Reliability requirements*
- *Safety requirements*
- *Maintainability and serviceability requirements*
- *Packaging requirements*
- *Transportation requirements*
- *and others*

Life-cycle considerations thus clearly increases the number of different requirement categories and many of these types of requirements are frequently encountered both in related literature and in industrial practice. Conversely, other authors like Suh (1990) have highlighted and focused on the fundamental differences between functional requirements and constraints. His Axiomatic Design Theory views functional requirements as descriptions of functions that the product should be able to perform to satisfy specific needs and possess particular properties such as decomposition and independence. For instance, "the system shall be able to make the vehicle move forward" is an example of a functional requirement for the powertrain system of a vehicle. This requirement can be then decomposed on sub-functional requirements for different sub-systems and components that will be working together to fulfill this higher level requirement. Constraints are viewed as bounds to what can be considered an acceptable design solution and normally are not clearly decomposable (Suh, 1990). They are usually related with bounds on size, capacity, weight, materials, cost, geometric shapes or interfaces.

A different taxonomy is the one proposed by Pohl (2010), who classifies requirements according to the type of software **artefact** being considered. His taxonomy divides requirements into three main types:

- *Goals* are stakeholder intentions regarding the objectives, properties and use of the system.
- *Scenarios* are concrete examples of situations satisfying or failing to meet goals. A scenario typically defines a sequence of steps and interactions that are needed to satisfy a goal and describe how these steps interact with the system's context.
- *Solution-oriented requirements* are requirements comprising the functional and behavioral properties of the system, quality requirements and constraints.

Pohl's taxonomy thus defines three sub-categories within solution-oriented requirements. Functional requirements describe functionalities that the system provides to its users; quality requirements define properties for the system, components or functions, like availability, efficiency, flexibility, integrity, reliability, robustness, etc; and constraints are

viewed as organizational or technical conditions that restrict the system's configuration (Robertson and Robertson, 2006).

Other authors have also traditionally distinguished between functional and non-functional requirements (Kotonya and Sommerville, 1998). These authors define non-functional requirements as the "overall qualities and attributes of the resulting system", including any constraints on "interfaces, quality, resources or time-scales". Moreover, according to Kotonya and Sommerville, non-functional requirements include performance, interface, operational, resource, safety, portability, quality and reliability or maintainability requirements. This view is however controversial and has triggered disagreements in literature. For instance, Pohl (2010) strongly recommends not using the term non-functional requirement in practice, arguing that non-functional requirements either are underspecified functional requirements or quality requirements.

## **2.3 The management process**

Various authors have defined and presented process models for requirements management. Kotonya and Sommerville (1998) define in their book that the process model is a "simplified description of a process", which is normally presented from the author's particular perspective. The following section reviews the process models presented across a wide range of literature where the topic has been discussed. A summary corresponding to the author's view is then presented in the following section.

### **2.3.1 Process models**

Pahl and Beitz (2007) define within engineering design literature the process of establishing a requirements list after the product planning phase of the product development process, without formally naming it. They defined essentially a sequential process consisting of five main stages [Pahl and Beitz (2007) p. 146]:

- The *identification stage* where requirements are identified according to market demands and the needs of particular market segments.

- The *definition stage* where requirements are defined, recorded, refined and ranked using check lists and scenario analysis.
- The *evaluation stage* which considers the circumstances under which requirements have to be met and distinguishes between demands and wishes.
- The *decision stage* which constructs and presents the requirements list formally into a table format that includes the user, the project name, the requirement's priority, the responsible person, the date of last change and the version of the requirement.
- And the *update stage* where requirements are changed, amended or extended. Pahl and Beitz state that not all requirements are known at the beginning of the design process and the requirements list is often changed during later development stages.

Their book mentions additionally that any change to the requirements list is responsibility of the chief designer and it should follow a "formal change management procedure", but no details of such process are presented (Pahl and Beitz, 2007). Another product design book by Otto and Wood (2001) arranges the requirements process into four stages: requirements compilation, conflict evaluation and resolution, requirements verification and change evaluation. Otto and Wood divide requirements into functional requirements and constraints similarly to Suh (1990) and propose an additional distinction into demands and wishes as Pahl and Beitz (2007).

A more formalized framework has been however presented by Fiksel and Hayes-Roth, who defined specifically requirements management as the "process of creating, disseminating, maintaining and verifying requirements" during the full life-cycle of the product (Fiksel and Hayes-Roth, 1993). Their work was subsequently followed by (Tseng and Jiao, 1999), who represented requirements management according to the process model illustrated in Figure 2.2 and defined it as the *iterative process* of performing:

- **Requirements elicitation** in which customer needs are identified using any methodology designed to understand the voice of the customer.
- **Requirements analysis** which is when customer needs are mapped into explicit

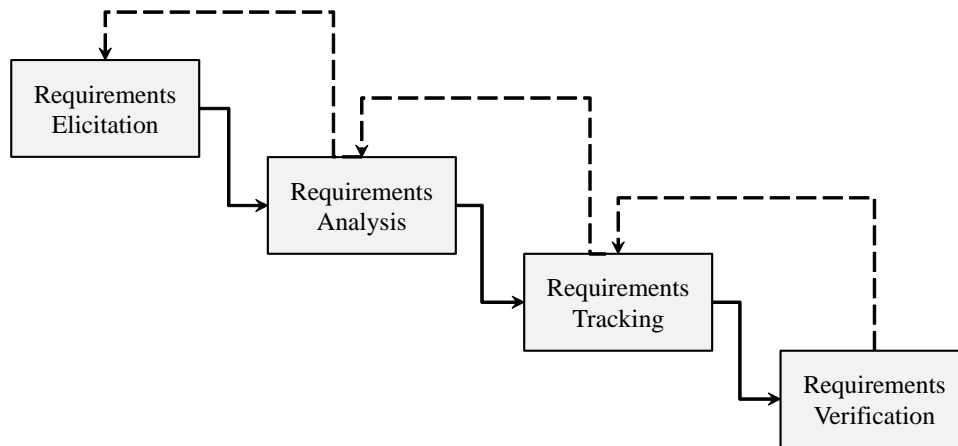


Figure 2.2: The requirements management process model of Fiksel and Hayes-Roth (1993). The illustration is based on the description of Tseng and Jiao (1999).

requirements that can be used by engineers and computer programs during the subsequent design stages.

- **Requirements tracking** in which requirements are continuously negotiated and changed within the design team in order to resolve conflicts and changes in objectives.
- **Requirements verification** which is when it is determined if the current design solution complies with the set of requirements in place at that time.

In a more recent contribution, Jiao and Chen (2006) revisited the model of Fiksel and Hayes-Roth and proposed a three-step requirements management process model comprising requirements elicitation, requirements analysis and requirements specification (Jiao and Chen, 2006). They defined requirements specification as the concrete definition of requirements in the functional domain of the product, including the continuous negotiation and resolution of conflicts which was previously defined in requirements tracking. Therefore, this requirements management process model is not fundamentally different but rather a variant of the practice illustrated in Figure 2.2.

In the software area, Kotonya and Sommerville define requirements engineering as the "structured set of activities which are followed to derive, validate and maintain a system's requirements document" [(Kotonya and Sommerville, 1998), p. 9]. These authors view requirements management as a *spiral* process – represented in Figure 2.3 – composed



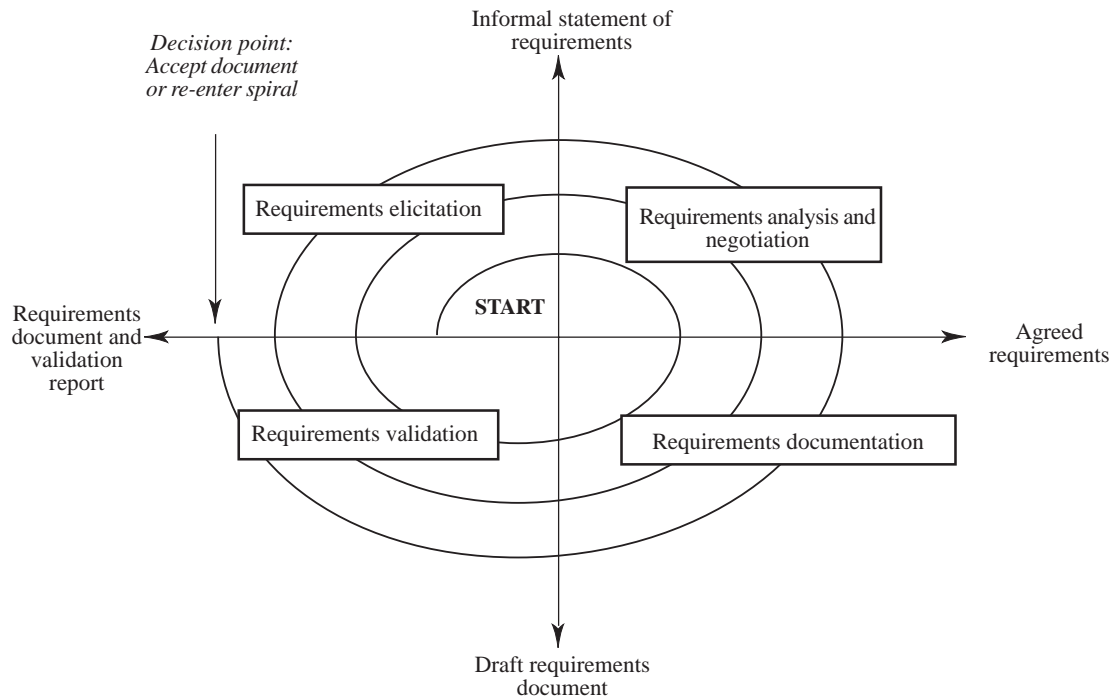


Figure 2.3: The spiral requirements engineering process model [(Kotonya and Sommerville, 1998), p. 35].

of four main activities:

- **Requirements elicitation** where requirements are discovered from stakeholders.
- **Requirements analysis and negotiation** where requirements are analyzed in detail and negotiated with all stakeholders, leading to requirements prioritization and conflict resolution.
- **Requirements documentation** where agreed requirements are documented using natural language and diagrams.
- **Requirements validation** where consistency and completeness of the requirements documents is checked.

Kotonya and Sommerville state this process is highly iterative since feedback between the activities is needed and repetition occurs until an acceptable solution is found or until business pressure drives the process to an end. The spiral process model also clearly implies that requirements change is *expectable*. The process of "identifying, analyzing, costing and implementing changes" to the system's requirements is defined as **requirements change**

**management** by Kotonya and Sommerville and represented in a three-step sequential process model that includes:

- *Problem analysis and change specification* which is where the identified problem in requirements is analyzed and changes are proposed.
- *Change analysis and costing* where the proposed changes are analyzed considering how many additional requirements are affected and the overall change cost is determined.
- And *change implementation* which is when requirements documents are amended or new versions are issued and consistency checks are repeated.

Figure 2.4 illustrates this change management process model proposed by Kotonya and Sommerville, showing that the change analysis and costing stage is mainly concerned with identifying requirements directly and indirectly affected by a change request and estimating the costs of implementing the resulting proposed changes. Finding the list of indirectly affected requirements requires knowledge about *traceability*, i.e. knowledge about dependencies. Change implementation stops at the modification of requirements documents and therefore this requirements change management process model does not cover solution implementation nor controls its quality.

In another major and more recent reference within the software field, Pohl (2010) defines requirements engineering as a "cooperative, iterative and incremental process" aiming to understand all requirements, achieve sufficient agreement with stakeholders and document them. Pohl categorizes requirements into goals, scenarios and solution-oriented requirements, as explained in Section 2.2. His requirements engineering process model includes three core activities: elicitation from stakeholders, documentation and negotiation to resolve conflicts. In addition, Pohl defines several management activities supporting the core process which include requirements prioritization, requirements traceability, configuration and change management, activity planning and control and observation of the context exterior to the organization for change detection (Pohl, 2010).

Pohl also distinguishes clearly between configuration and change management activities,

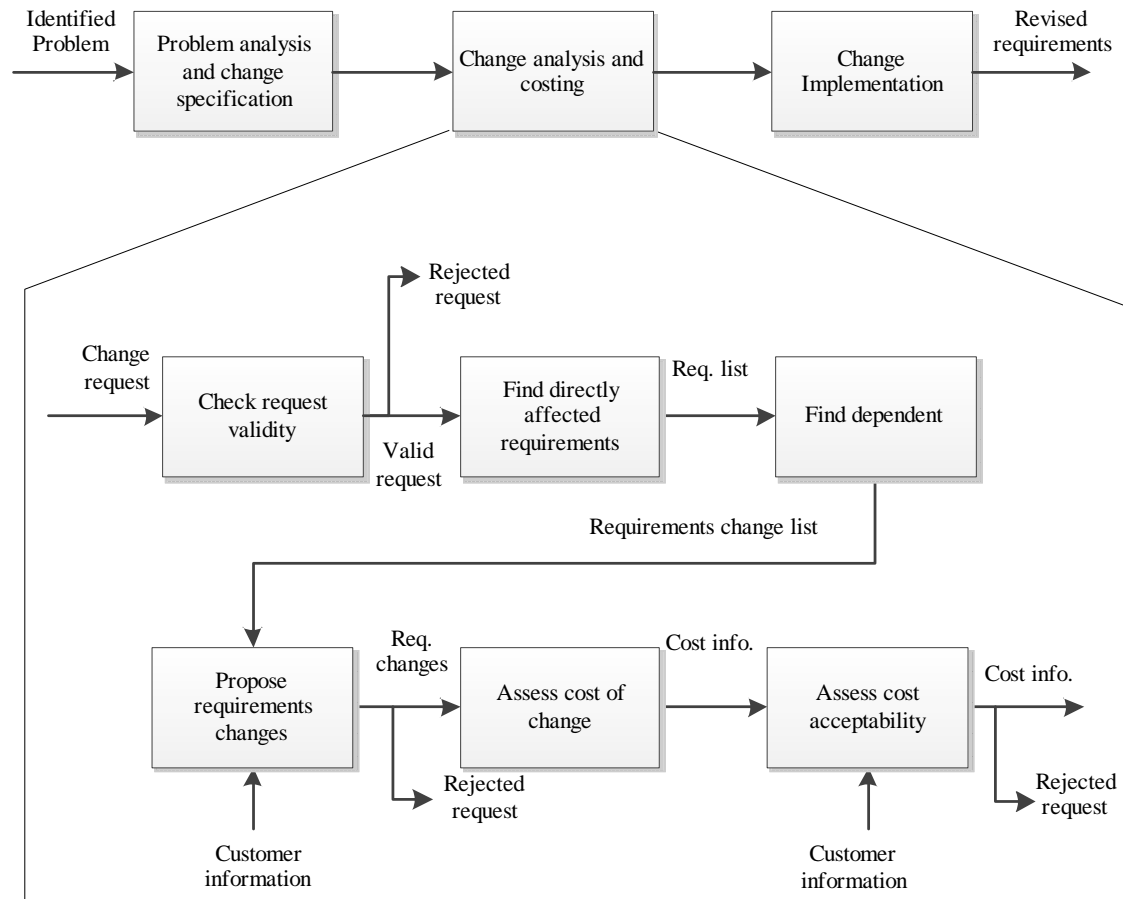


Figure 2.4: Activities in the requirements change management process model of Kotonya and Sommerville (1998), adapted from illustration in p. 124.

defining configuration management basically as managing the evolution of document versions over time to ensure coherence is maintained. Conversely, systematic change management is concerned with consistently integrating requirements change into existing requirements and Pohl (2010) defines a five-step change management process model which is represented in Figure 2.5 and comprises:

- **Change request classification** where requests are categorized into corrective change, adaptive change or exceptional change.
- **Impact analysis** where the effort needed to integrate the change request into the requirements and into the system is estimated. In order to determine the total effort, all requirements and software objects affected by a change request must be identified.
- **Change request evaluation** where the cost and benefits of the each change are

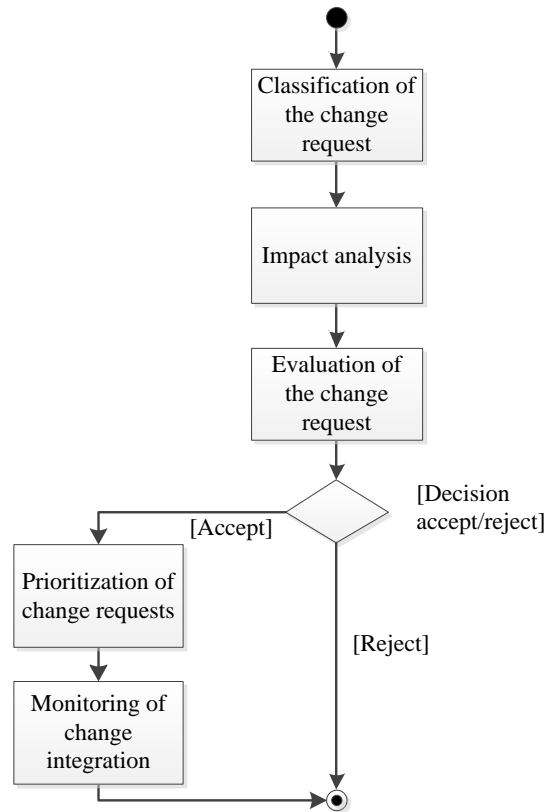


Figure 2.5: The requirements change management process of Pohl (2010), adapted from illustration in p. 656.

evaluated.

- **Change request prioritization** where change requests are bundled according to their priority and assigned to a change project or to a particular system release.
- **Change integration monitoring** where the status of each change request during realization is monitored.

Furthermore, Pohl refers that a *change control board* or a *change steering committee* should be responsible for taking decisions on requirements changes and prioritization. Such board or committee is usually composed of people representing the main stakeholders in the project and people with the technical knowledge and experience required to take the decision.

The books of Pohl and Kotonya and Sommerville show that requirements management is a major concern in software engineering and best practices are detailed in this domain. In fact, a well-known industrial survey performed by the Standish Group showed that

almost half of project failures in software development were caused by poor requirements management (Standish, 1995). The same study showed that requirements change alone accounted for 12% of total amount of project failures and constituted the third single largest cause of failure in software development. Industrial data thus supports the importance of requirements management within software development.

In addition, requirements management is also of great importance for systems engineering, due to the reasons explored in Chapter 1. INCOSE (2006) states that the success of projects aiming to develop complex systems or systems over long time-scales depends upon the execution of a formal requirements management process. This process is defined as the "collection, analysis and validation of requirements with all the communications and negotiations inherent to working with people" [INCOSE (2006), p. 177].

This definition relates well to the activities defined in the process models represented in Figures 2.2 and 2.3. However, the whole process is significantly more difficult during complex system development since it must be performed *across* the systems's functional hierarchy. Chapter 1 referred that the process starts at the beginning of a new project through the capture of high-level stakeholder needs: required system mission, functions, performance, operational capabilities, maintenance expectations and cost and schedule constraints, for instance. These high-level requirements are then used to drive an iterative design loop where the system's functional architecture and physical design solution are developed and verified against its requirements.

New functional requirements are *derived* from a verified system solution and *allocated* to the next level of the system's decomposition. The derived and allocated requirements become the set of higher-level requirements at the next level – e.g. the sub-system level — and guide a similar iterative design loop at that level (NASA, 2007). The process is then repeated until the lowest level of the system's hierarchy is reached. Iterations of both requirements and design solutions can also occur across system levels during integration activities to ensure the overall compliance and consistency.

Figure 2.6 represents these interactions between requirements and design processes during space systems development. NASA (2007) defines the activity shown in Figure 2.6 of

deriving and allocating requirements from a higher system level to a lower one as *requirements flow down*. This activity – which is not included in the process models described in engineering design and software development literature – is key for a successful systems engineering approach during complex systems development.

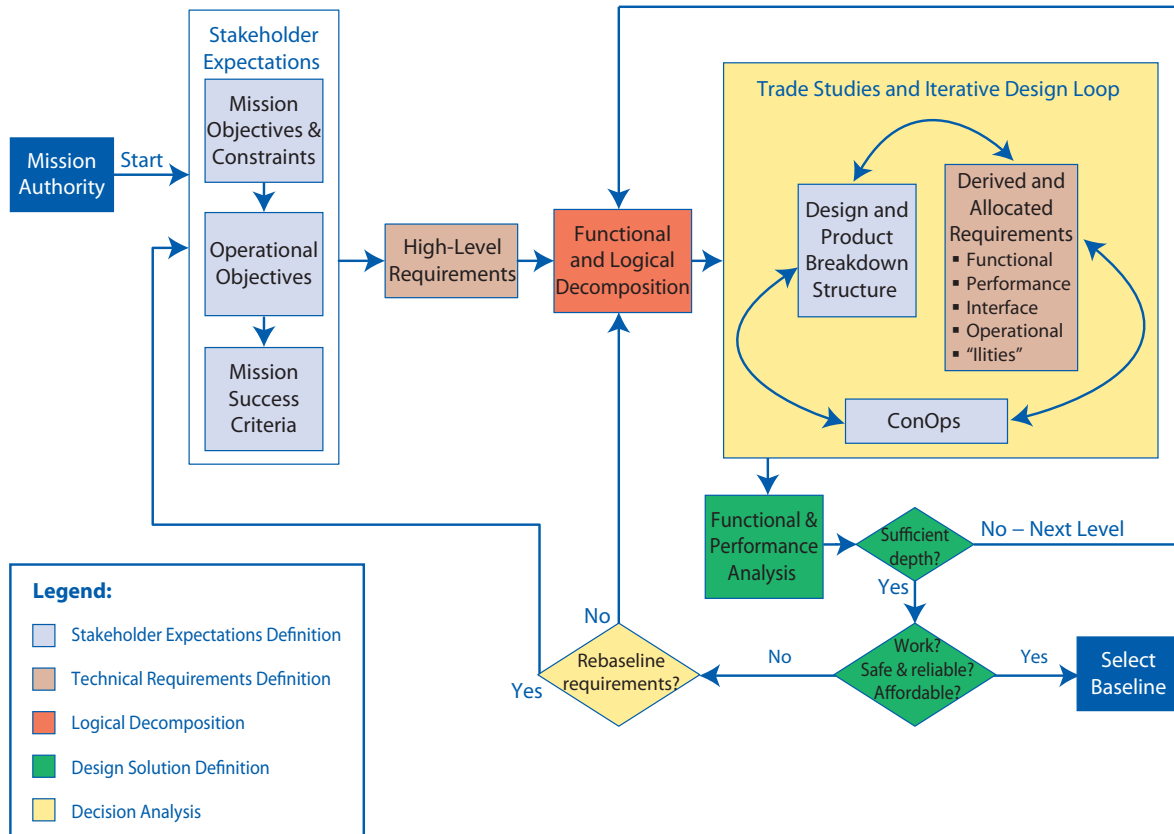


Figure 2.6: Overview of the relationships between the requirements and system design processes during space systems development from NASA (2007), p. 31.

### 2.3.2 Summary

The previous review shows that requirements management process models appear across different literature bodies and have been presented from different perspectives. The author's view – synthesized from this literature review – is that requirements management encompasses two main processes: a *core requirements engineering process* and a *support change management process*.

The core engineering process includes five activities: requirements elicitation, requirements analysis and negotiation, requirements documentation, requirements validation and

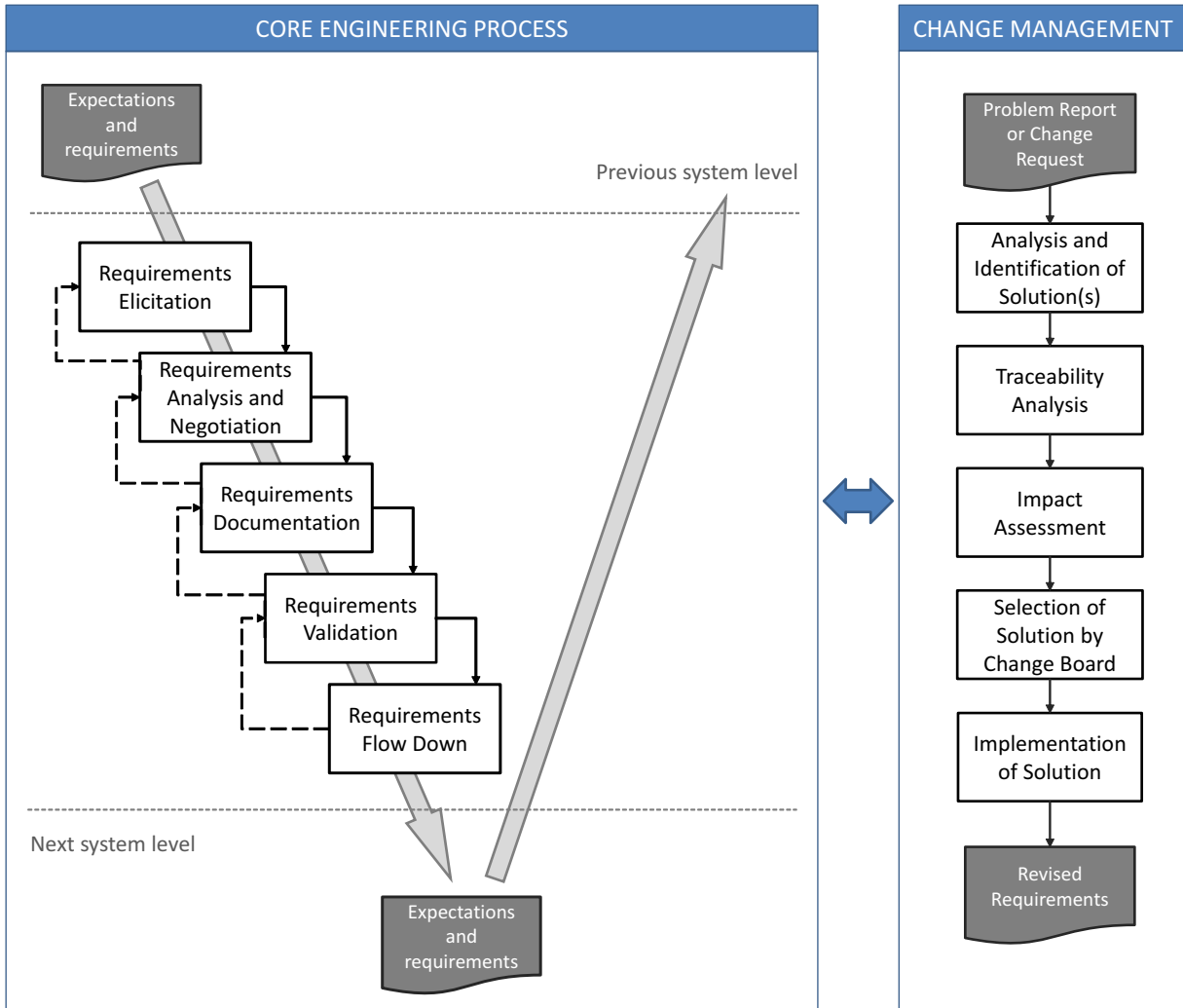


Figure 2.7: This thesis' representation of the requirements management process model for industrial practice.

requirements flow down. During complex system development, the core requirements engineering activities are performed sequentially and iteratively at all levels of the system's functional hierarchy, following the top-down systems engineering approach. This view of the core engineering process model is represented in Figure 2.7. In addition, the core engineering process is supported by a change management process since requirements evolve during time and there is the need to incorporate updates in a systematic and controlled approach. This thesis attempts in Figure 2.7 to synthesize the common change management activities found in all models. The process starts with the analysis of a problem report or a requirements change request. Following the identification of possible solutions, an analysis of requirements traceability and the impact of change is performed. A change control board then selects and approves a solution among alternatives, which is

subsequently implemented and leads to a revised set of requirements for the system.

## **2.4 Support methods and tools**

Following the author's summary of the main aspects of the requirements management process, the subsequent sections review and explore a wide range of methods and tools presented in literature which intend to support specific activities in the process, such as requirements elicitation, prioritization, traceability analysis or change impact assessment.

### **2.4.1 Requirements elicitation**

Capturing and translating stakeholder needs into objective statements that can guide engineers is critical for product development success. Because of that, requirements elicitation has motivated a wide range of approaches and investigations targeting to understand or improve the capture process. Furthermore, the amount of literature produced during the last couple of decades on this topic led also various authors – e.g. Christel and Kang (1992), Shaw and Gaines (1996), Coughlan and Macredie (2002), Jiao and Chen (2006) and Zhang (2007) – to provide reviews, comparisons and discussions of the different approaches.

Due to the abundance of prior reviews, this thesis discusses key aspects of various elicitation methods based on the recent work of Zhang (2007), which is summarized in Table 2.1. Zhang classified elicitation methods into four main types: conversational, observational, analytic and synthetic. Conversational methods use verbal communication with stakeholders to capture their needs and problems and include interviews, workshops or focus groups and brainstorming. Conversely, observational methods relying on ethnographic studies or protocol analysis attempt to observe at distance how stakeholders perform activities in their own environment in order to capture requirements, particularly those which are normally difficult to articulate verbally by people.

In addition, requirements can also be captured from prior systems developed in the organization (requirements reuse), from documentation released by stakeholders – such as



regulations or standards – (documentation studies) or from the experts’ knowledge using techniques such as laddering, card sorting or repertory grid which provide structured approach to expert knowledge capture. Table 2.1 shows that previous approaches were classified as analytical methods used in requirements elicitation. Finally, Zhang refers also interactive sessions with stakeholders to discuss scenarios, storyboards or prototypes together with the combination of conversational and observational methods as requirements capture approaches. These were coined as synthetic methods (Table 2.1) and may be employed at any stage of the development process, in opposition to other elicitation methods which are more suited to be deployed during the early stages of product development.

Table 2.1: Methods supporting requirements elicitation (Zhang, 2007).

Type	Method(s)	Description
Conversational	Interviews	Requirements are understood from prepared questions and discussions about the system with individual stakeholders.
	Workshop / Focus group	Requirements are captured or reviewed from a focused and intense debate conducted with a group of stakeholders.
	Brainstorming	Requirements are gathered in an unstructured list from fast and spontaneous discussions with stakeholders.
Observational	Social analysis / Observation / Ethnographic study	Observer spends time embedded in the stakeholder’s environment to capture requirements using detailed observation of the stakeholder’s activities.

Continued on next page

Table 2.1: Continued.

Type	Method(s)	Description
	Protocol analysis	Requirements are captured from the stakeholder while he is executing a task and concurrently thinking out loud about it.
Analytic	Requirements reuse	Requirements are identified from legacy systems or other systems within the same product family.
	Documentation studies / content analysis	Requirements are captured from available documentation concerning the desired system, such as policies, standards or market information.
	Laddering	Requirements are captured from a stakeholder through the organization of his knowledge into a hierarchical structure, such as a tree diagram.
	Card sorting	Given a set of domain objects / attributes in cards, requirements are identified through sorting them into groups according to the criteria used by the stakeholders.
Analytic	Repertory grid	Requirements are captured from a process of identification of domain entities and rating of attributes characterizing those entities.

Continued on next page

Table 2.1: Continued.

Type	Method(s)	Description
Synthetic	Scenarios / Passive storyboards	Requirements are identified from a session with stakeholders describing a sequence of actions and events that can occur while the system is performing a task.
	Prototyping / Interactive storyboards	Requirements are elicited from stakeholders through interactions with partial versions of the system.
	JAD / RAD sessions	Joint Application Development (JAD) and Rapid Application Development (RAD) search elicitation from group sessions and workshops moderated by a facilitator, using visual aids, brainstorming and top-down analysis.
	Contextual inquiry	Combines interviews, observation and prototyping for requirements elicitation in applications where the user interface design is critical for the system.

### 2.4.2 Requirements prioritization

The author's literature review showed also that several methods have been developed to support engineers ranking and classifying requirements during the analysis, negotiation and documentation stages of the process (Figure 2.7). Prior work includes simple prioritization techniques, the Wieger's method and the Cost-Value approach:

- **Simple prioritization** is based on the subjective judgment of stakeholders and/or

on one or two simple qualitative criterion such as risk, necessity or time criticality (Pohl, 2010).

- The **Wieger's method** is an analytic prioritization method based on four criteria: benefit, penalty, cost and risk (Wieggers, 2010). Each stakeholder or group of stakeholders assigns a numeric score to each of the previous attributes using, for instance, a 1–10 numeric scale. Each criteria may also be assigned a weight –  $W_b, W_p, W_c, W_r$  respectively – according to its relative importance to the stakeholders. The requirements,  $R_i$ , priority is then computed from:

$$Priority(R_i) = \frac{Benefit(R_i) \times W_b + Penalty(R_i) \times W_p}{Cost(R_i) \times W_c + Risk(R_i) \times W_r} \quad (2.1)$$

- And the **Cost-Value approach** is a pairwise comparison technique proposed by Karlsson and Ryan (1997) based on the Analytical Hierarchy Process (Saaty, 1990). Assuming there are  $n$  alternatives, Karlsson and Ryan uses this method to compare pairs of requirements considering its relative *cost* and *value* which are fed into a  $n \times n$  matrix. The total relative cost and total relative value of each requirements is determined computing the principal eigenvectors of the matrix and communicated using a cost-value diagram such as the one reproduced in Figure 2.8. The cost-value diagram then supports the prioritization of requirements during the management process.

### 2.4.3 Traceability analysis

Traceability denotes the degree to which a relationship can be determined between objects having a "predecessor-successor" or "master-subordinate" kind of relationship (IEEE, 1990). Gotel and Finkelstein (1994) defined requirements traceability as "the ability to describe and follow the life of a requirement, in both a forwards and backwards direction". The ability to retrieve dependencies to predecessors and successors of a requirement has been defined respectively as pre and post-traceability (Gotel and Finkelstein, 1994). Fig-

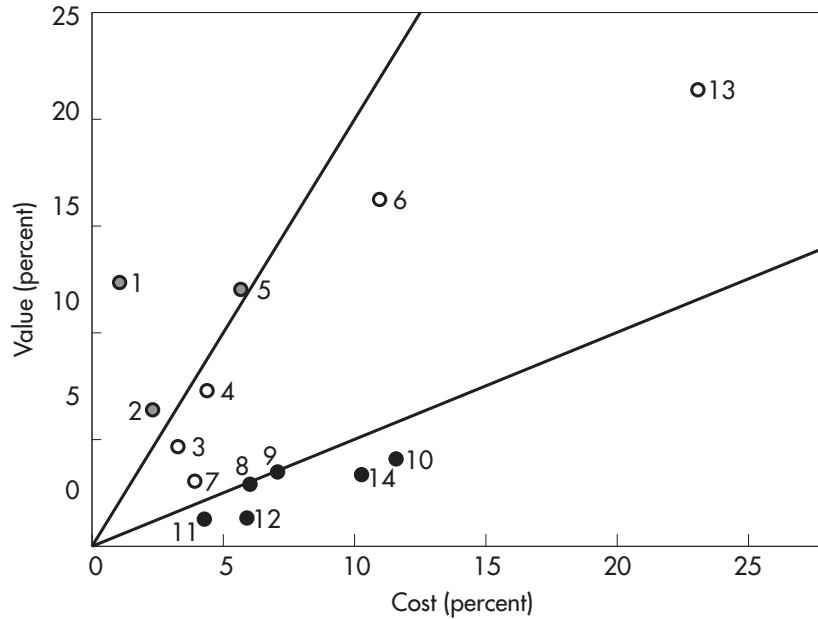


Figure 2.8: The cost-value diagram obtained by Karlsson and Ryan (1997) when prioritizing requirements during projects. Individual requirements are identified by numeric values (1-13).

ure 2.7 showed that traceability analysis is a key activity during requirements change management. Pohl (2010) also advocates that requirements traceability information can support various other activities during system development, namely quality assurance, re-engineering, risk management and accountability. Because of that, specific techniques and approaches have been developed to support traceability analysis, namely:

- **Textual referencing** which is the simplest form based on writing a textual annotation pointing out a dependency relationship of any kind to another system entity. An example is "R2-17: For selecting the trip destination, the navigation system shall display the last then trip destinations. [based on  $\rightarrow$  R1-17]" (Pohl, 2010), p. 614.
- **Cross-referencing schemes** which is a technique enhancing or customizing textual referencing based on special syntax schemes that can be automatically read by software applications (Jackson, 1991).
- **Hypertext-based referencing** which uses hyperlinks to document traceability typically in electronic versions of requirements documents and may store differ-

**DEPENDS FROM** →

← **DEPENDS ON**

	R1	R2	R3	R4	R5	R6
R1			X			X
R2	X			X		
R3		X			X	
R4						
R5	X	X				
R6		X		X		

Figure 2.9: A simple requirements traceability matrix (Kotonya and Sommerville, 1998)

ent types of relationships such as replacement, satisfaction, refinement, conflict or derivation dependencies (Kaindl, 1993).

- **Information flow-based models** which use abstract "box" and "arrow" graphical models representing the flow of information to document requirements traceability.
- and **Matrix-based models** which use matrices to document, present graphically and analyze dependencies. Kotonya and Sommerville (1998) present for instance *traceability tables* to document relationships between requirements and its predecessors or successors. Figure 2.9 shows an example which places X marks in the cells where some kind of dependency relationship exists between two requirements. By reading down a column, we see all dependencies of the requirement on that column; conversely, by reading across the row, we see the dependencies originated from that same requirement.

The traceability table is in fact another version of the dependency matrix concept. The original idea of storing dependencies or interactions of a design, system or project activities belongs to Steward (1965). The original matrix-based approach was named as Design Structure Matrix (DSM). It has also received many other names since then, including dependency structure method, dependency structure matrix, n-square matrix or design precedence matrix.

### 2.4.4 Change impact assessment

Change impact assessment is another key activity during requirements change management, as mentioned in Section 2.3 and depicted in Figure 2.7. Impact assessment is normally executed by identifying the entities directly and indirectly affected by a requirements change. Knowledge about the direct dependencies of a requirement with the other entities – *direct impact* – and knowledge about how a requirements change propagates through indirect dependencies and affects other entities – *indirect impact* – are thus fundamental to conduct a change impact assessment. Two methods are commonly used in direct change impact analysis:

- **Expert elicitation** where interviews and other knowledge elicitation techniques are used to retrieve the expert’s understanding about the list of dependent entities, the probability of requirements change and the impact of that change on each of the dependent items. The interviewer may ask the expert to quantify the impact of the requirements change either qualitatively or quantitatively. In a qualitative assessment, the expert may be asked to categorize the change impact based on a pre-defined defined qualitative scale: low to high for instance. Conversely, the expert can be asked to provide a quantitative value for that change impact based on a prediction of man-hours needed for the requirements change implementation, for instance.
- **Traceability analysis** techniques such the ones described in Section 2.4.3 are also commonly used in direct impact analysis since they store direct dependencies between entities. However, the methods’ scope varies considerably. Textual and hypertext-based referencing methods are typically limited to documenting dependencies between requirements and demand low effort. On the other hand, information flow models, graphs and dependency matrix methods can provide dependencies between requirements and between requirements and other system entities but require a greater building effort.

Expert elicitation methods have been used for instance in an extensive empirical study about requirements change impact in software development (Lindvall, 1997). Lindvall demonstrated also in his doctoral dissertation that interviewing colleagues who are perceived as knowledgeable was the method preferred by developers in practice but most actually doubted that expert elicitation is capable of finding all affected system entities. Techniques used in traceability analysis are a mature technique and well covered in textbooks (Kotonya and Sommerville, 1998; Lamsweerde, 2009; Pohl, 2010). If dependency relationships are consistently recorded from the beginning of projects and they include links from requirements to requirements and/or other system entities, then traceability can support change impact assessment in an effective manner. Morkos et al. (2012) provide a recent application of such techniques to predict requirements change impact and propose the use of higher order DSMs to support designers understanding propagation of a requirements change into other requirements.

Additionally, several other techniques have been previously developed to facilitate indirect change impact assessment:

- **Automatic propagation methods** relying on the idea that change propagation in a system can be computed *automatically* using a mathematical algorithm if direct dependencies between elements are known beforehand. In other words, the algorithm searches for all change propagation paths of a given *initiating change* and predicts the total accumulated impact of that change on the system defined.
- **Heuristic methods** relying in relationships deduced from theory or experience about how simple measures of change propagation are related with the overall indirect impact of a change.

A well-known example of automatic propagation algorithms is the Change Prediction Method proposed by Clarkson et al. (2004) to compute the impact of engineering changes – defined as “alterations made to parts, drawings or software that have already been released” (Jarratt et al., 2011) – during product design. Clarkson et al. require historical records or expert knowledge to populate two types of DSMs: a direct likelihood matrix, representing the probability of a change in one component leading to a change in another;



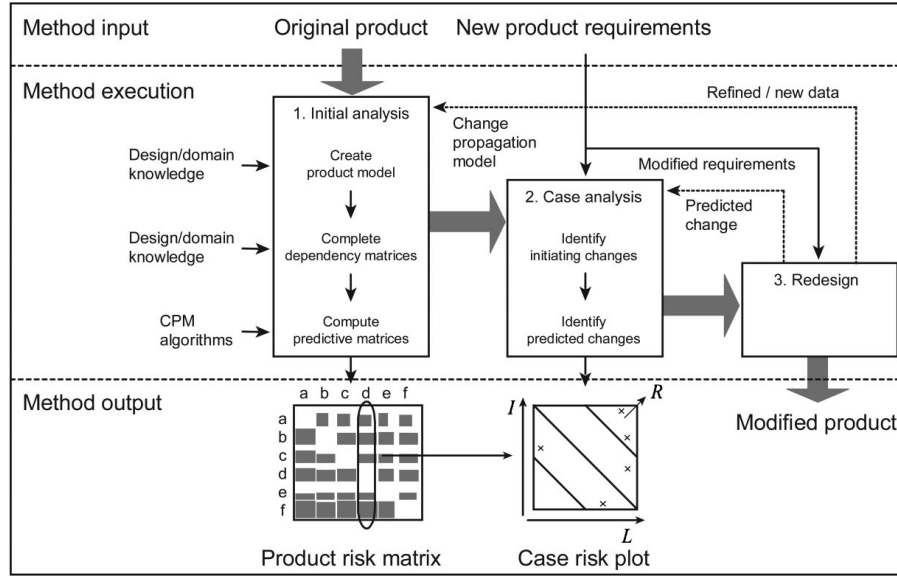


Figure 2.10: Overview of the Change Prediction Method proposed by Clarkson et al. (2004).

and a direct impact matrix, representing the amount of rework needed to implement that change. The algorithm then predicts the total impact of the change from the whole propagation tree (Clarkson et al., 2004). The method has subsequently been applied in several industrial studies, including change impact prediction in helicopter design (Clarkson et al., 2001), diesel engine design (Jarratt et al., 2004) and jet engine fan design (Koh and Clarkson, 2009). Figure 2.10 shows the main features of the Change Prediction Method. Other methods based on DSMs have also been proposed by Carrascosa et al. (1998), Rutka et al. (2006) and Lemmens et al. (2007). All methods can be applied to various types of engineering changes, including the ones initiated by a change in product requirements.

The main advantage of automatic propagation is its predictive capabilities based on simple mathematics. There are however several limitations. It requires extensive commitment from experts to build DSMs and to elicit the values for the direct probability and direct impact of change (Keller et al., 2006). Moreover, experts think that "matrix filling is an unpleasant task" as observed by Koh and Clarkson (2009) during a jet engine fan design case-study. And the number of propagation paths grows exponentially with the number of entities in model, which limits the method's scalability. Because of that, the propagation

depth is normally limited to 3 or 4 levels and models with more than 50 elements are not recommended (Clarkson et al., 2004).

Conversely, heuristic methods can be used to reduce the effort of performing change impact analysis at the expense of some accuracy. Several heuristics have been proposed based on graph theory and graph measures to support the probabilistic evaluation of change propagation (Keller et al., 2006):

- The probability of change propagating between two nodes is inversely proportional to the shortest path length, defined as the number of dependencies connecting them.
- The probability of change propagating between two nodes is directly proportional to their edge connectivity, defined as the number of different paths connecting the nodes.
- The probability of change propagating between two nodes is directly proportional to the number of common neighbors.

Although it seems theoretically possible to derive similar heuristics to support designers quantifying the indirect impact of change, no formulation has been presented in literature so far to the author's knowledge. Table 2.2 summarizes the key strengths and limitations of the various approaches and methods that can be used to perform requirements change impact assessment.

## **2.5 Empirical contributions**

Research aiming to understand and improve the requirements management practice has also included empirical studies, although this type of research is traditionally difficult to accomplish and publish due to the intellectual property and confidentiality concerns that surrounds the industrial environment. Empirical studies are also a major source of knowledge in this practice area and thus this subsection aims to provide a comprehensive review of prior research contributions of this nature.

Recognizing that establishing requirements is critical for the design activity and thus a

Table 2.2: Summary of strengths and limitations of various approaches supporting change impact assessment.

Approach	Strengths	Limitations
Expert elicitation	<ul style="list-style-type: none"> <li>• Simple and practical</li> <li>• Requires low effort</li> <li>• Good for direct impact assessment</li> </ul>	<ul style="list-style-type: none"> <li>• Difficult for experts to assess probabilities</li> <li>• Accuracy can be questioned</li> <li>• Difficult for experts capturing all affected items</li> <li>• Poor for indirect impact assessment</li> </ul>
Traceability-based	Referencing: <ul style="list-style-type: none"> <li>• Good for direct impact assessment</li> <li>• Good accuracy if dependencies are consistently recorded</li> </ul> Flow and matrix-based: <ul style="list-style-type: none"> <li>• Good for indirect impact assessment</li> </ul>	Referencing: <ul style="list-style-type: none"> <li>• Limited to impact analysis in the requirements domain</li> </ul> Flow and matrix-based: <ul style="list-style-type: none"> <li>• Requires high effort to build and maintain</li> <li>• Difficult to interpret and communicate large models</li> </ul>
Automatic propagation	<ul style="list-style-type: none"> <li>• Predictive capabilities</li> <li>• Good for automatic indirect impact assessment</li> <li>• High potential to discover unforeseen relationships due to change propagation</li> </ul>	<ul style="list-style-type: none"> <li>• Requires high commitment from experts to build and maintain</li> <li>• Overall accuracy depends of elicitation quality of direct probabilities of change and direct impact</li> <li>• Cannot resolve large models, limited scalability</li> </ul>
Heuristics	<ul style="list-style-type: none"> <li>• Require lower effort than automatic change propagation methods</li> <li>• Good for indirect impact assessment</li> </ul>	<ul style="list-style-type: none"> <li>• Require previous knowledge about most relevant propagation paths</li> <li>• Lower accuracy than automatic change propagation methods</li> </ul>

central issue for design research, Chakrabarti et al. (2004) researched the process and methods used by designers to identify, generate and apply requirements during design activities. Their experimental study with four designers showed that requirements identification is highly connected with solution generation and detailing and the understanding, evaluation and remembering of requirements during the design process is dependent upon the use of appropriate requirements management support tools (Chakrabarti et al., 2004).

A major empirical study containing a comprehensive overview of requirements management and of issues arising in practice in the Swedish automotive industry has been previously reported by Almefelt et al. (2006). These authors used archival research, personal interviews and workshops/ seminars to understand and reconstruct qualitatively the requirements management process during the development of a passenger car cockpit. The case-study approach revealed many interesting findings about the difficulties arising in practice, such as the misunderstanding of requirements among stakeholders, complexity in requirements management across interconnected systems, managing supplier requirements for outsourced systems and components and handling requirements evolution and change during projects.

A survey of requirements engineering capabilities in small-to-medium companies is also part of prior knowledge (Michael et al., 2007). This empirical study revealed that requirements management capabilities are normally not well-defined in these companies and there is a general lack of understanding about how to capture customer needs and manage requirements change during product development. The importance of comprehending change management practices has also captured the attention of Sudin and Ahmed (2009), who examined an empirical data set of 271 specification change reports of an aero-engine. This study found that most change reports occurred during the manufacturing and testing phase due to the implementation of product improvements and correction of design errors.

Subsequent empirical studies addressed requirements management support for mechatronic and embedded systems development in the automotive industry (Bergsjö et al., 2010; Ćatić and Malmqvist, 2010). Particular problems in the development of these systems were analyzed using mainly qualitative research methods, such as difficulties arising from informal requirements management and problems in the management of high-level requirements spanning the product which cannot be delegated to specific engineering teams (Bergsjö et al., 2010). A recent case-study performed by Sudin and Ahmed-Kristensen (2011) aimed also to understand the decision-making and organizational factors influencing the requirements change management process in practice. The topic was explored

qualitatively at a product development consultancy company and showed that change emerges as a result of internal activities and external factors arising from technology or market changes. In addition, an extended empirical research performed by Vianello and Ahmed-Kristensen (2012) – partly based on the work of Sudin and Ahmed (2009) – examined a large quantity of engineering change reports across two products. This recent study showed that most change requests occurred during the manufacturing phase and confirmed that requirements change was one of the key causes of engineering changes during the development phase (Vianello and Ahmed-Kristensen, 2012).

The relationship between requirements change and engineering change has also triggered related research contributions. Engineering change accounts for modifications to parts, drawings or software that have been previously released during the development process and various authors have investigated its causes. Pikosz and Malmqvist (1998) identified specification changes, design problems found during prototype development or production and product quality issues as key reasons. Fricke et al. (2000) reported changing or incorrect requirements, product innovation, error correction and change propagation as major causes of engineering change. Eckert et al. (2004b) categorized the sources of change either as emergent (arising from the product) or initiated (arising from an outside source) and discussed case-study results showing that changes in customer and certification requirements, product innovations and design error correction identified during development, testing, fabrication or service are important causes of engineering changes. The automatic change propagation methods mentioned in Section 2.4.4 – such as the Change Prediction Method (Clarkson et al., 2004) – have also been used in various recent studies to understand the effects of changes in requirements on the product’s architecture (Hamraz et al., 2012; Koh et al., 2012). Requirements change has thus been established as a major source of engineering change in a broad range of empirical research contributions.

## 2.6 Review of progresses

### 2.6.1 Opportunities for research

This chapter provided a comprehensive literature review with the intent of putting this dissertation into the context of prior work around the topic of requirements management. This review showed various definitions of the term requirement, various taxonomies, multiple process models across different literature bodies and a wide range of methodologies developed to support particular requirements management activities. Furthermore, previous empirical investigations have attempted to contribute to a better understanding of the management problems arising in industrial practice.

However, the review showed also that the causes of requirements change in practice have only been investigated qualitatively (e.g. Almefelt et al. (2006)) or discussed as a source of engineering change (e.g. Pikosz and Malmqvist (1998), Fricke et al. (2000), Eckert et al. (2004b), Sudin and Ahmed (2009), Vianello and Ahmed-Kristensen (2012)). Considering prior research contribution, this thesis argues that an *in-depth* and *quantitative* investigation to the root causes of requirements change in practice has not been reported yet. The author's understanding is also that prior work has not been able to access a substantial industrial data set which can be used to characterize how and why requirements evolve during the development of complex systems. The protection of confidential information and proprietary rights may explain the academic difficulty of performing such an investigation. But understanding how requirements evolve and what are the causes of change in detail is likely to result in improvements to the requirements management practice of organizations developing complex systems. Consequently, the gap identified as a result of the literature review presented in this chapter was considered by the author as a major opportunity for original research contributions.

### 2.6.2 Progress regarding research questions

Aside from providing a comprehensive literature review, placing this dissertation into context and identifying research opportunities, this chapter made also research progresses

to derive an answer to research question Q1-a. The author elaborates below the answer to this question:

**Q1-a** *What are the best practices in requirements management?*

The core requirements management best practice includes the process of eliciting, negotiating, documenting and validating requirements during the system's life-cycle. Elicitation identifies the needs of stakeholders in their own voice using conversational, observational, analytic or synthetic methodologies. Analysis and negotiation solves conflicts and leads to requirements prioritization. Documentation establishes agreed requirements and validation allows the overall consistency to be checked. During the design and development process of large technical systems, the core process includes also requirements decomposition and flowing down from higher levels to lower levels of the system's functional hierarchy in order to guide engineering teams working with sub-systems or components, for instance.

Best practices include also the management of changes to requirements during the whole life-cycle of the system. Change management practices complement the core requirements engineering activities and include the evaluation of problem reports or change requests, identification of possible solutions, traceability analysis to understand the range of entities affected by a change, assessment of the impact of that change and implementation of the solution selected by a company Change Board.

## 2.7 Summary

This chapter provided a review of prior research contributions around the topic of requirements management and included various perspectives arising from several bodies of literature, such as engineering design, software engineering and systems engineering. Since the scope of literature around the topic is very large, the author has focused on key references that influenced the research presented in this dissertation. In a short summary, the chapter's key conclusions are:

- The term "requirement" relates to qualities or constraints identified from multiple sources that the product must satisfy.
- Different taxonomies for requirements have been presented according to the criteria of relative importance, source of capture, product life-cycle considerations and type of system artefact.
- Best requirements management practices include a core engineering process and support change management activities. The core process encompasses the elicitation of requirements from stakeholders, analysis and negotiation, followed by requirements documentation and validation. The change management process supports the core engineering process with analysis of problem reports and change requests, assessment of the impact of change and implementation of a selected solution leading to a revised version of requirements. The author's view of the key aspects of the best practice has been summarized in Figure 2.7.
- Various methods and tools have been developed to support the engineering and change management processes. Table 2.1 showed that requirements elicitation can be performed by practitioners using a wide range of approaches, which encompass conversational, observational, analytic and synthetic methods. Requirements prioritization occurring during analysis, negotiation and documentation can be executed using expert judgment, the Wieger's method and the Cost-Value approach. Analysis of traceability may be accomplished from the use of textual referencing, cross-referencing, information flow models and matrix-based models. And Table 2.2 showed that expert elicitation, traceability-based methods, automatic change propagation and heuristics can be used by practitioners during change impact assessment.
- Empirical studies have also provided major insights about the requirements management practice and comprise experimental studies investigating processes and methods used by designers, industrial case-studies providing qualitative findings about practical problems and challenges, company surveys to understand capabilities and



analysis of change reports arising from development projects.

- There is a gap in literature arising from lack of detailed and quantitative knowledge about how requirements evolve and change during the course of complex product development and it is argued this constitutes a major opportunity for novel academic contributions.

In conclusion, the comprehensive literature review presented in this chapter allowed an answer to the research question Q1-a (*What are the best practices in requirements management?*). Chapter 3 explores the research gaps and opportunities arising from this literature review, which triggered a large-scale empirical study performed by the author to the root causes of requirements change during the development of complex systems.



# Chapter 3

## An empirical study of root causes of change

Becoming truly effective at requirements engineering and management is challenging for organizations designing and developing complex systems. Chapters 1 and 2 argued that the process is inherently difficult for several reasons. A very large number of requirements typically need to be captured, documented and change managed at different levels of the product's physical and functional hierarchy (Weber and Weisbrod, 2003). Multiple external and internal stakeholders are continuously involved in negotiation (Pohl, 2010). And the engineers performing the process must act and take decisions in an environment containing technical uncertainty, semantic ambiguity (Tseng and Jiao, 1999) and imperfect communication. Correctly and completely capturing external needs into internal requirements for design and development activities, or deriving and flowing down new requirements from existing ones at higher levels of the product hierarchy, is highly dependent on skills, experience and organizational capabilities.

This chapter draws on a large-scale empirical study performed by the author during the course of eight months, which were spent on-site at Rolls-Royce Derby investigating the evolution of requirements and the root causes of change during the development of gas turbines for aerospace applications<sup>1</sup>. It is argued that this study contributes to a better

---

<sup>1</sup>This chapter is largely based in the material previously presented in Fernandes et al. (2015). Reprint is made with permission.

understanding of requirements change during complex product development and complements prior literature discussed in Chapter 2. The current chapter proceeds in five main sections. It begins by a presentation of the study's objectives. Secondly, the research setting surrounding the organization and the case selected is presented. The approach implemented by the author during the course of the study is then explained. Fourthly, findings from the case-study are presented and analyzed in detail. Finally, improvement guidelines for management resulting from the research are discussed. The current chapter ends with a discussion of the research progresses made and a summary of conclusions arising from this investigation.

### **3.1 Objectives**

The aim of this chapter is to bring forward a large-scale empirical study about the causes of requirements change during the product development process of large technical systems. The study was performed at Rolls-Royce, a global firm which designs, manufactures and maintains gas turbines such as the one illustrated in Figure 1.3 for its customers.

The author engaged with Rolls-Royce in research since this company is continuously looking for improvements to requirements management and, generally speaking, to all of its systems engineering processes. The empirical study focused on the requirements evolution during the development process of a previous aerospace gas turbine project that took place during the course of 6 years. The author was allowed to collect and analyze the project's requirements database, which contained more than 700 system requirements. These had evolved through more than 1000 changes released during the development process. The assessment of the causes of change was performed by the engineers *actually* involved in the project. In addition, the author was able to reconstitute the requirements engineering process dynamics and related it with the quantitative results about the causes of change. The quantitative and qualitative results allowed Rolls-Royce engineers to identify and dissect a number of management guidelines aiming to improve requirements management practices further. All these findings are presented and discussed in this chapter.

The author's investigation proceeded with three main objectives. The first was to discover

what were the root causes of requirements change during complex product development. The second was to quantify how the causes of change vary during different phases of the development process. And the third objective was to report how much change is generated externally to the organization compared to internally driven change. These goals derived logically from the research questions Q1-b, Q1-c and Q1-d presented in Section 1.4 and from the research opportunities detected from the literature review (Chapter 2).

## **3.2 Research setting**

### **3.2.1 Organizational context**

Rolls-Royce follows a product life-cycle management approach and a gate-review development process similar to the ones reported in systems engineering literature (Blanchard and Fabrycky, 2006) and illustrated previously in Figure 1.1 (Chapter 1). Figure 3.1 represents the particular process followed by Rolls-Royce, showing that this organization has formally divided it into seven main stages. Innovation and Opportunity Selection (Stage 0) is the initial phase where business opportunities for a new product are identified, a preliminary business case is constructed and initial concepts are developed. Stage 1, Stage 2 and Stage 3 defined in Figure 3.1 correspond, respectively, to the Conceptual Design, the Preliminary Design and the Detailed Design stages described in Section 1.1. Product and Continuing Service Support (Stages 4 and 5) refer to the entry-into-service of the product and include the stabilization of production and supply chain processes, operational support to customers and introduction of product improvements also described in Figure 1.1. Finally, Stage 6 corresponds to end of life disposal, following the decision of ceasing production due to product obsolescence.

System design and development starts formally at Rolls-Royce during Stage 0 and ends at Stage 3 exit, which is when the product enters officially into service (Figure 3.1). An advanced and separate multi-disciplinary engineering unit – called Future Programs Engineering (FPE) – conducts the design and development activities during Stage 0 and

Stage 1, namely requirements capture, concept exploration and preliminary definition of a system design solution. These stages are marked by frequent interactions with the customer, which regularly sends a Request-for-Information (RFI) about the current design solution status that FPE must respond to. In order to respond to a customer RFI, FPE often needs more detailed assessments of sub-system or component design solutions and thus involves various internal Supply Chain Units (SCUs) – e.g. Turbines, Compressors, etc – through bid requests.

Stage 1 exit is normally marked by a formal Request-for-Proposal (RFP) that FPE has responded to and which resulted in a concept being selected by the customer. At that time, an Integrated Program Team (IPT) is formed that comprises a chosen group of design and manufacturing engineers that spans all SCUs. This large team is responsible for proceeding with the full concept definition during Stage 2 and the detailed design of all sub-systems and components during Stage 3. Because of that, it is usually broken down into smaller teams responsible for single sub-systems, components or even just specific design disciplines, such as the aerodynamic design of a single component. The IPT is led by a Whole Enterprise Definition (WED) team, which includes a team of five to ten Chief Engineers responsible for the whole product. The WED team directs and plans the technical work performed by the IPT to produce the design solution and it is responsible by the integration of sub-systems and components.

Within each stage and between stages, business and technical reviews occur, as shown in Figure 3.1. Each review intends to validate that the new product complies with all the business and technical requirements that have been specified, for an acceptable level of risk. The main technical reviews between Stage 0 and Stage 3 include the Concept Review (CR), the Preliminary Design Review (PDR), the Critical Design Review (CDR) and the Production Readiness Review (PRR). The CR ensures that the explored concept complies with requirements and assesses the development plan. The PDR is held when the analytical and experimental verification of the chosen concept is complete and confirms key functionalities and attributes. The CDR approves the design and manufacturing definitions at a level of detail which is considered appropriate to start development

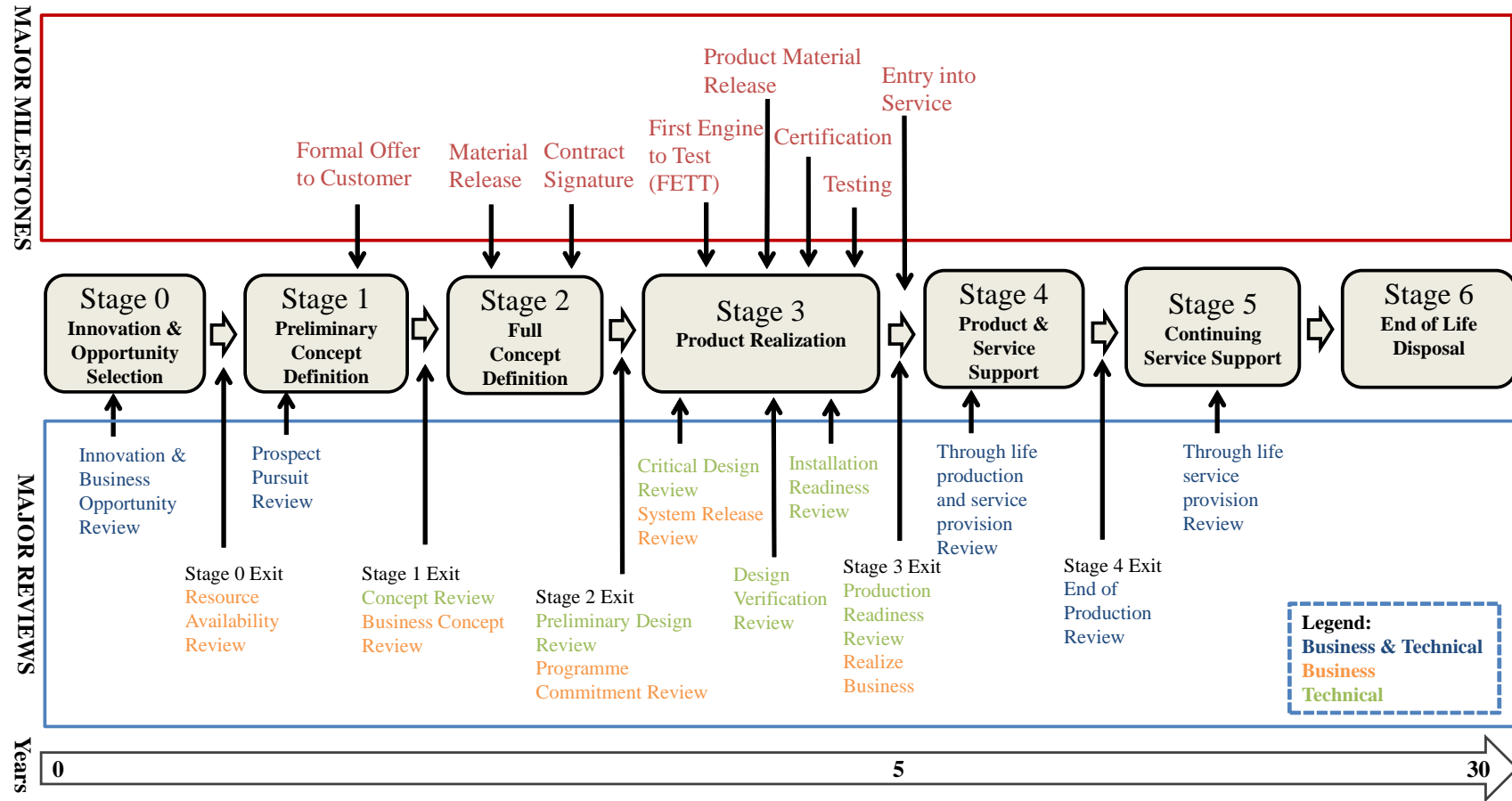


Figure 3.1: The product's introduction and life-cycle management process defined by Rolls-Royce, together with major reviews and milestones. Adapted from Rolls-Royce (2011).

hardware manufacture. Finally, the PRR reviews the final product definition against all requirements.

These reviews occur not only at system level, but also at sub-system and component level and with increasing levels of technical detail, covering the design and manufacturing definition as well as the project plan. Change actions usually emerge from reviews and are implemented by the IPT to address and correct problems that have been identified.

### **3.2.2 Company requirements management practices**

Rolls-Royce uses requirements-driven processes (INCOSE, 2006) to develop its products systematically. The exploratory research – reported in Chapter 1 – allowed the author to investigate also the company’s requirements management practices. This investigation revealed that the requirements management process implemented in the company follows the main activities recommended in state-of-the-art literature reviewed in Chapter 2, namely stakeholder identification, requirements elicitation, analysis, prioritization and balancing, requirements documentation and flow down. Additionally, it also provides a basis to ensure the execution of validation and verification activities.

Figure 3.2 illustrates the iterative process defined at Rolls-Royce for concurrently mapping requirements to solutions at different system levels. The general capture principle defined in the process found by the author is simple and scalable, following the systems engineering V-model (Blanchard and Fabrycky, 2006). The process is based on a top-down approach, starting in the beginning of any project at the highest level, through the capture of high level stakeholder needs into requirements for the whole system (Figure 3.2). When it is verified through evidence that the proposed design solution satisfies the requirements, key solution parameters are flown down to the next system level as requirements and the process is then repeated for each subsequent lower level (Figure 3.2).

The company’s requirements management process also standardizes the documentation landscape to be maintained by any product development project. The landscape of requirements and corresponding definition documents is represented in Figure 3.3. High-level stakeholders needs are initially translated into a Business Requirements Document



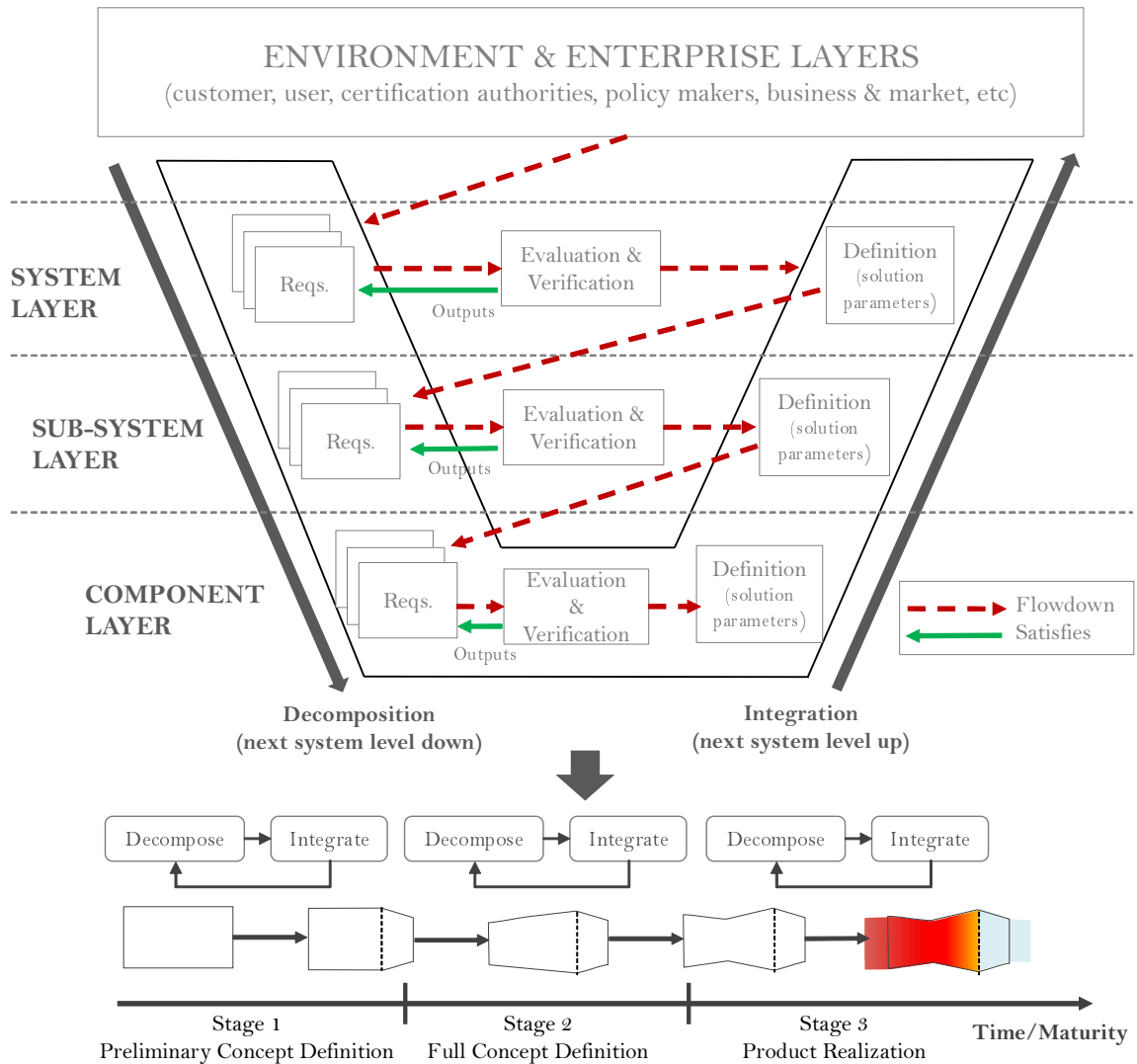


Figure 3.2: General principles in requirements engineering defined at Rolls-Royce, following the systems engineering approach to product development. Adapted from internal company sources.

(BRD) or cascaded directly to the Product Requirements Document (PRD), located at system level. The BRD contains also business needs elicited from internal stakeholders, such as the expected profit or production volumes. Some of the business requirements are cascaded to the system level, either directly or through the associated definition document. The PRD encompasses all the system requirements and the Product Definition Document (PDD) is created through design and subsequent evaluation and verification activities. The cascade of requirements from the system layer – both through direct flow down or from requirements derived from the solution’s definition – is then repeated for the sub-system and component levels. Figure 3.3 shows this process originates Sub-System

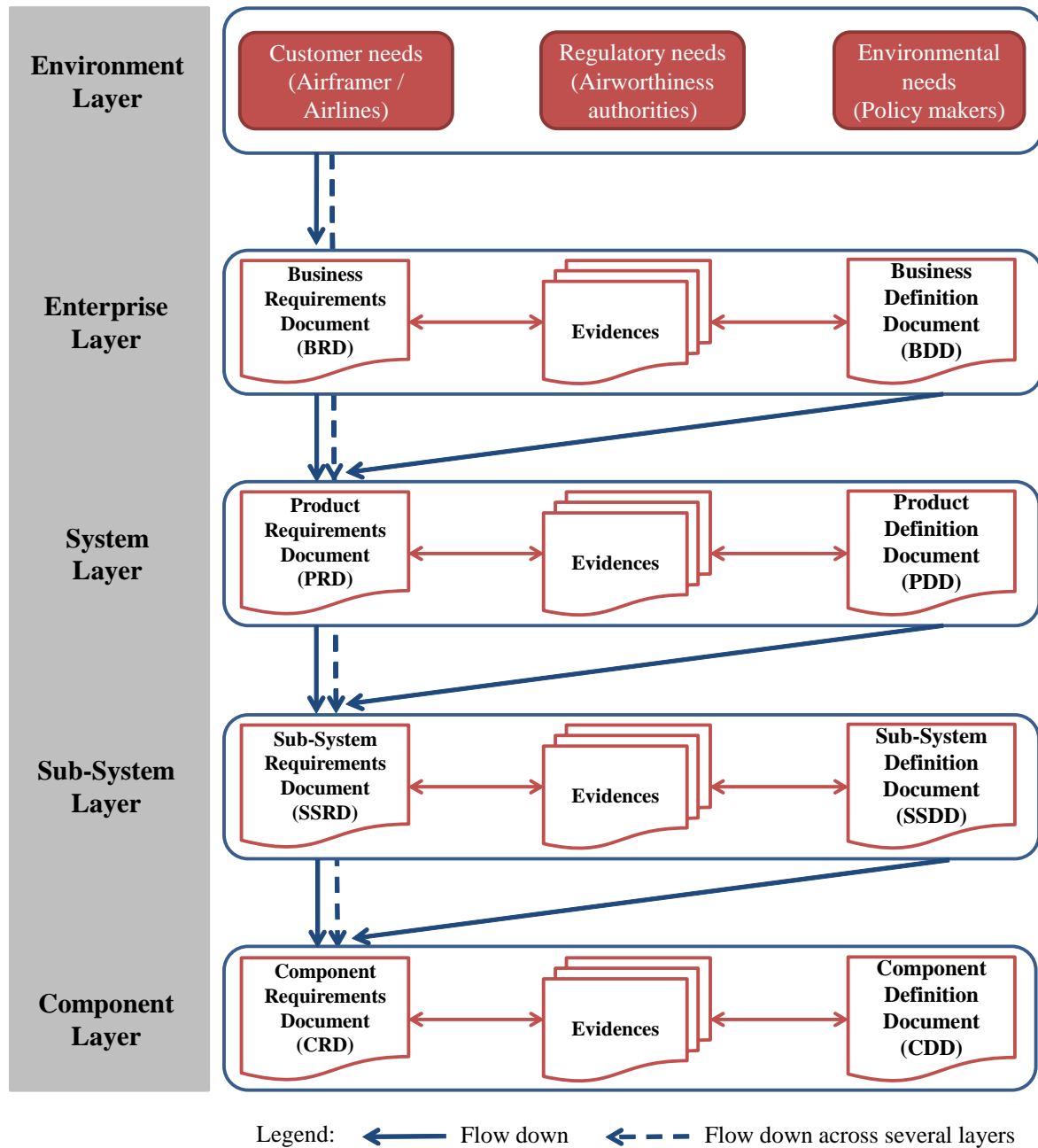


Figure 3.3: Landscape of requirements and definition documentation at Rolls-Royce. Adapted from internal company sources.

Requirements Documents (SSRDs) and Component Requirements Documents (CRDs) and the corresponding solution definition documents.

Furthermore, the author found that projects are constantly monitored to ensure that the best requirements management practices are being followed and to search for improvements. Training is provided to staff. For instance, training material supports engineers writing good requirements. The company's aim is that all designers are proficient

in requirement engineering and can work independently as requirements' authors. The role defined for systems engineering specialists in Rolls-Royce is then to support designers when needed, facilitating and driving requirements capture and understanding. For practical reasons, a smaller group of engineers may be appointed by the project as *requirements managers* at a particular system layer. These engineers work together with system, sub-system or component designers in requirements capture and are responsible for structuring and maintaining the requirements database during the project. In addition, Rolls-Royce has standardized methods and tools to conduct stakeholder needs analysis, resolve conflicts, populate and manage the requirements database and create and maintain the document landscape in product development projects.

### **3.3 Research approach**

The understanding about the previous research setting arose from the exploratory research performed by the author during the time spent on-site. In addition, the set of 14 semi-structured interviews performed to a sample of engineers and managers reported in Section 1.2 was also used to select the case to study. Due to higher availability of data and staff that had lived the project, the development of a particular gas turbine was selected for the investigation. This section presents the case studied and the research approach implemented by the author.

#### **3.3.1 The case studied**

The case studied was the recent product development process of an aerospace gas turbine at Rolls-Royce. The project's time frame is represented in Figure 3.4 together with key development milestones. The project departed from business conversations with the customer which resulted in the signature of a first Memorandum of Understanding (MoU) framing the development of a new system for a particular aerospace application. At that time, Rolls-Royce's advanced programs department was highly involved in concept development and selection.

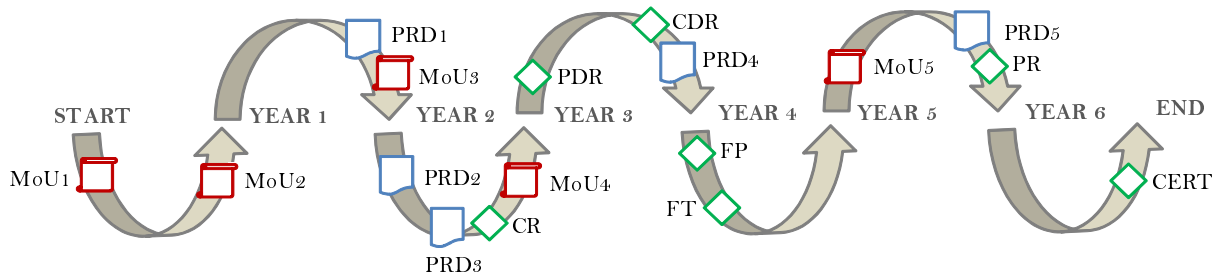


Figure 3.4: Time frame and main design and development milestones of the project investigated at Rolls-Royce. Legend: MoU - Memorandum of Understanding; PRD - Product Requirements Document; CR - Concept Review; PDR - Preliminary Design Review; CDR - Critical Design Review; FP - First Parts; FT - First Test; PR - Production Readiness; CERT - Certification.

The first PRD, containing the initial requirements for the *whole system*, was formally released one and a half years after the first business conversations with the customer. Two new versions of the PRD followed closely and came out during the second year of the project. Concept exploration, concept selection and preliminary design work lasted until the beginning of the third year, when a PDR took place. Several interaction loops with the customer had occurred in parallel (Figure 3.4).

Detailed design activities continued during the third year and the design was initially "frozen" after passing the CDR. Following the CDR, a fourth issue of the PRD was released. The first test of the new gas turbine occurred during the spring of the fourth year (Figure 3.4). Several development prototypes were then tested during an extensive validation and verification program lasting one and a half years. By the end of the fifth year, the initial program requirements for certification were delayed by the customer. Additionally, major changes in some of the customer requirements also occurred. Around the same time, a fifth issue of the PRD was formally released by the project. The final stages of testing and development that lead to certification occurred close to the end of the project's sixth year.

### 3.3.2 Reasons for selecting a case-study strategy

Yin (1984) states that a case-study approach is a method adequate for an empirical investigation intending to study a "contemporary phenomenon within its real-life context,

when the boundaries between phenomenon and context are not clearly evident and in which multiple sources of evidence are used". It is suitable for investigations intending to provide "cause-effect" relationships and for explanatory types of research, dealing with "how" and "why" research questions (Lindvall, 1997).

In this empirical study, it was not possible to separate the phenomenon of interest - the causes of requirements change - from its technical and organizational context. The nature of research questions Q1-b, Q1-c and Q1-d was also intrinsically explanatory. The author was interested in understanding how requirements evolve, how change is generated and how do the causes of change vary during the product development process. Explaining the causes behind observed changes in requirements is about understanding a cause-effect type of relationship.

All of the above supports the choice of a case-study strategy. Moreover, it is rare that scientific investigations are allowed to study such phenomena in an industrial environment. Because such phenomena are typically inaccessible, the case study selected for this investigation can be considered a revelatory case (Yin, 1984).

### **3.3.3 Methodology and data sources**

Archival research followed the initial stage of exploratory research. The author started with the collection of system requirements and an analysis of its evolution which was embodied in the changes performed during the course of the development of the new gas turbine. The belief was also that an investigation to the causes of requirements change would allow the author and Rolls-Royce to identify potential improvements to the requirements management process. Prior work performed at Rolls-Royce by Pickard et al. (2010) and Nolan et al. (2011) reported that most requirements change in control systems' development is internally-driven, but an in-depth and quantitative study focused on the development of the whole system had not been realized yet.

The overall methodology followed during the study is presented in Figure 3.5. The population of changes was collected from archival research in documents and from a commercial software tool – IBM's Rational DOORS – used in the company to conduct requirements

engineering and manage the requirements database. Each requirement contained a unique identification tag, which allowed following its evolution. The database provided also the author of the requirement, time of creation and time of change. Following an initial analysis of the database, the author decided to focus the study at *system level*, since the data was too large for a detailed study of all system layers within a reasonable research time frame.

After the data collection stage, a preliminary list of causes of change was defined by the author based on the observation of the requirements evolution. The company's requirements management software tool contained all records of changes, together with notes and comments from the engineers responsible for requirements engineering. It was thus possible to develop hypothesis about the root causes in many cases. As shown in Figure 3.5, the initial list was then submitted to a process of revision and validation.

The 4 requirements managers that had worked with the system's requirements and managed the changes during the course of the project were invited to participate in the research. Details about the engineers participating in the research are provided in Appendix B. They were initially interviewed during approximately 45 minutes each, using a prepared semi-structured guide. Interviews followed again the recommendations of Robson (2002). The interview guide focused on the history of the project's requirements engineering process and contained the proposed list of causes. The engineers were asked if the list was comprehensive and how they interpreted the definition of each cause of change. This step was performed to understand the capability of distinguishing in practice the different root causes and thus evaluate the risk of misclassification. Following this round of interviews, adjustments were made based on the feedback received to the list of causes and their definition to minimize that risk.

Analysis of the system requirements database revealed more than 1000 changes between *released versions* of product requirements documents ( $PRD_i$ ). The size of this population led the author to design a representative sample. Moreover, to ensure accurate and unbiased results, the assessment of each requirement change was performed by the engineer that had actually worked on it. The requirements managers had an inside knowledge

	Understand context	Collect data	Define causes	Design sample	Quantify causes	Capture improvement
Semi-structured interviews	✓		✓			
Archival research	✓	✓	✓		✓	
Statistical methods				✓	✓	
Expert knowledge elicitation sessions					✓	✓

Figure 3.5: Research approach and methods used throughout the course of the case-study.

about the full history of the requirement and had lived the project. Such knowledge cannot be easily captured from archival research. It had to be *elicited*. The engineers thus determined themselves the causes of change in the sample defined in the following stage of the research (Figure 3.5).

The author prepared a specific template containing the new version of the requirement, the previous version, the type of change and the list of causes of change, which was available from a drop down menu. This template is also shown in Appendix B. Prior to each session, the author prepared a small set of hypothesis about the cause of each requirement change from the history recorded in the requirements management software tool, to generate discussion with the engineers. This facilitated knowledge elicitation, making the requirements managers justify their assessment by remembering the history of the requirement, the sequence of events and the interactions with other individuals. This strategy intended to increase the confidence in the results. During the following stage of the research, each engineer was then invited to determine the cause for each of the changes that he/she had executed in separate working sessions with the author.

The *proportion* of each cause of change was then quantified in statistical terms from the sample assessed by the requirements managers. As shown in the overall approach (Figure 3.5), a series of group working sessions were subsequently organized with the engineers to discuss the statistical results characterizing the requirements evolution and the causes of change during the project. This allowed capturing from the engineers involved in the research a series of management guidelines that could be or were being implemented at Rolls-Royce to further improve requirements engineering practices.

Throughout the course of the investigation, multiple sources of information have been used, from archival research, to semi-structured interviews and expert knowledge elicitation. Furthermore, the research methodology used of both *quantitative* and *qualitative* methods. Case-study construct quality (Yin, 1984) has remained a concern, as demonstrated by the multiple validation loops performed and the triangulation of information from archives and documentation with the knowledge elicited from the requirements managers.

### **3.3.4 Defining causes of change**

The methodology described in the previous section led the author to the causes of change presented in Table 3.1. Although the list presented was derived in the context of the particular project studied at Rolls-Royce, this thesis argues it is applicable to most projects since it covers a comprehensive set of issues experienced in practice during requirements management:

- The issue of externally driven change is included in customer, certification and business induced changes.
- Issues in requirements capture that may cause change are covered through missing requirements, incorrect capture, incomplete capture and incorrect or ambiguous language. Other capture related problems are described by requirements redundancy, decomposition and merger.
- Problems in the cascading process are described by incorrect and incomplete flow down.
- Inability to fulfill requirements or to verify them - change due to unachievable or unverifiable requirements - represents typical issues found during requirements validation and verification activities that often lead to change.
- And difficulties encountered during traceability management are also included in Table 3.1 as potential causes of change.



Table 3.1: Causes of change agreed with requirements managers at Rolls-Royce.

Cause of change	Definition
The customer changed the requirement	A change in customer needs, such as a change in the customer specifications, caused the requirement to change
Regulation or certification changed	A change in regulation or certification demands caused the requirement to change
Business requirements changed	A change in business needs caused the requirement to change
The requirement was missing	Although needed, the requirement was not previously captured and that caused the change
The capture was incorrect	The requirement was not correctly captured, due to wrong content or language, and that caused the change
The capture was incomplete	The requirement was captured but content was missing and that caused the change
The language was incorrect or ambiguous	Content was correctly captured but the requirement's language or syntax was incorrect or unclear
The flow down was incorrect	The requirement was cascaded incorrectly from an upper level, e.g., misplaced, and that caused the change
The flow down was incomplete	Although needed, the requirement was not cascaded from an upper level and that caused the change
The requirement was decomposed	The requirement was decomposed into multiple requirements and that caused the change
The requirement was redundant	Awareness that the requirement was not needed, that content was repeated or that over-specification occurred, caused the change
The requirement was merged	Awareness that the requirement should be merged into another requirement caused the change
The requirement was unachievable	Awareness that the requirement was not achievable caused the change
The requirement was unverifiable	Awareness that the requirement could not be verified or validated caused the change
Traceability links or references changed	The content and the language were correct, but a traceability link or a reference mentioned in the requirement changed (e.g., updated, removed, etc)
Other	The cause doesn't fit any of the above and it was elicited by the expert's own words

Specific reasons not fitting in the above categories were grouped under the "Other" category (Table 3.1). Considering the range of causes of requirements change covered in Table 3.1, the author argues it complements existing literature on the topic (Kotonya and Sommerville, 1998; Robertson and Robertson, 2006; Pohl, 2010).

### 3.3.5 Sample design and representativeness

As previously referred, the statistical quantification of the requirements evolution was performed based on a representative sample. The study consisted in having the requirements managers from the project determining the cause of each change from a list of reasons. It thus equals to test each change against each possible cause and to verify whether the cause applies or not. According to sampling design methods (Lohr, 2010), this case can be approximated by the case of a *proportion study*, where the researcher is interested in determining the proportion  $p$  of the population that has some characteristic  $y_i$ . This variable is binary, taking  $y_i = 1$  if individual  $i$  (a change in this case) has the characteristic (a possible cause) and  $y_i = 0$  if it doesn't. Lohr (2010) shows that for a finite population of size  $N$  the sample size required for a chosen confidence interval  $z$  and margin of error  $e$  is a function of the proportion  $p$ :

$$n \geq \frac{Np(1-p)}{p(1-p) + \frac{e^2}{z^2}(N-1)} \quad (3.1)$$

The expression has a maximum when  $p=0,5$  which is recommended when no prior knowledge exists. Sample sizes were computed for a level of confidence of 95% and a margin of error of 10% in our study.

One of the main concerns of sampling design methods is to ensure that the samples designed are representative of the population. In this regard, the population of changes could be naturally divided into sub-groups of additions, modifications and removals, according to the *type of change*. Conversely, the population is also naturally grouped around *types of requirements* - operational requirements, performance requirements, mass requirements, etc.

It was concluded through observation that large variations existed in the population and

OBSERVED POPULATION				
	Additions	Modifications	Removals	TOTAL (%)
Operational	A	B	C	8%
Performance	D	E	F	14%
Mass	...	...	...	3%
Safety	...	...	...	...
....	...	...	...	...
Reliability	...	...	...	6%
Cost	W	X	Z	2%
TOTAL (%)	45%	31%	24%	100%
TOTAL SIZE				N

↓

SAMPLE DESIGNED				
	Additions	Modifications	Removals	TOTAL (%)
Operational	a	b	c	8%
Performance	d	e	f	14%
Mass	...	...	...	3%
Safety	...	...	...	...
....	...	...	...	...
Reliability	...	...	...	6%
Cost	w	x	z	2%
TOTAL (%)	45%	31%	24%	100%
TOTAL SIZE				n

Figure 3.6: Design principles used to arrive to a representative stratified sample. Values are illustrative only.

thus a simple random sample would be highly unrepresentative. A stratified sampling design method was consequently selected. In addition, stratified samples were designed using the proportional allocation method (Lohr, 2010). Stratified samples were created to respect the two types of strata identified in the population, representing the distribution of change across the population according to the type of change *and* the type of requirements. The design principles used to engineer a representative stratified sample are illustrated in Figure 3.6. Maximum differences in sample proportions relatively to the population were approximately 1%, due to rounding to the closest integer inside strata. Simple random sampling was then used inside the different strata to select the required number of changes from the population into the sample.

### 3.4 Presentation and discussion of findings

Similarly to an engineering change (Jarratt et al., 2011), a requirements change refers to a modification performed to requirements previously *released* in the organization. The current section analyzes and discusses the evolution of system requirements. This evolution was investigated at the time of *release* of the Product Requirements Document (PRD). After initial capture, new released versions of the PRDs correspond to points in time where a major revision loop was finished and a complete set of updated system requirements was agreed in the project. Therefore, when referring to the amount of changes at a point in time, such value should be interpreted as the *accumulated amount* between two consecutive released versions ( $PRD_i$  and  $PRD_{i+1}$ ).

#### 3.4.1 Evolution of active requirements and types of change

Requirements evolve essentially through *modification* or *removal* of existing requirements and *addition* of new requirements. These can be thus considered different *types* of change. Lam and Shankararaman (1999) proposed the number or proportion of modifications, additions and removals in a given reporting period of a project as indicators of the requirements volatility (to distinguish between stable and unstable requirements) and the system's maturity. The author applied these concepts during this research to characterize and understand the evolution of the system's requirements in the project studied.

In addition, this thesis defines *active requirements* as the number or proportion of system requirements actually being *used* at a given point in time. The number of active requirements is given by the balance between the requirements modified, added, removed and left unchanged relatively to the previous reporting period or document release:

$$Active = Nochange + Modified + Added - Removed \quad (3.2)$$

The evolution of the project's system requirements is presented in Figure 3.7, built from the five major releases of PRDs that occurred during the development of the new gas turbine at Rolls-Royce. The time-series found shows that the number of active system

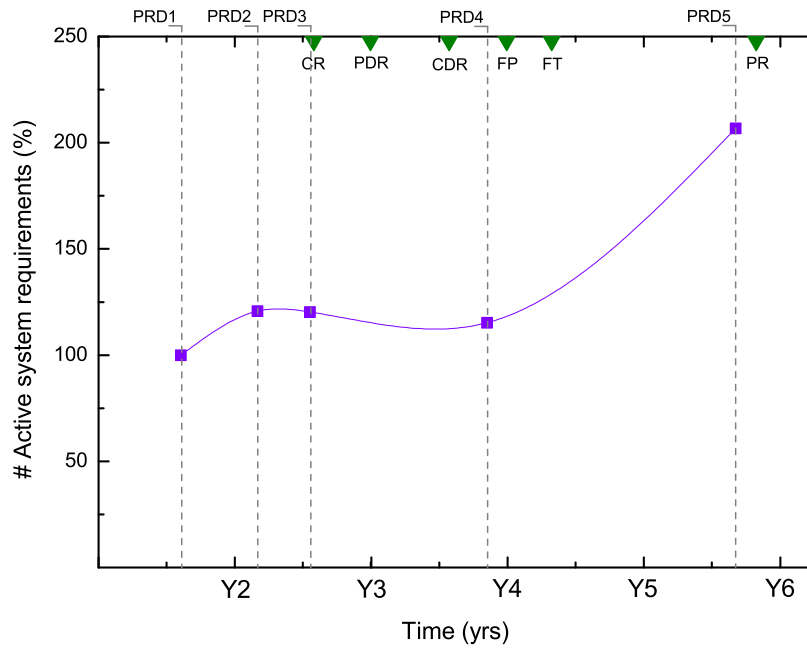


Figure 3.7: Evolution of active system requirements throughout the course of the development of a new gas turbine. Figures correspond to values normalized by the number of requirements captured at the initial release of the Product Requirements Document.

requirements roughly *doubled* during the course of the project, when compared with the number at the time of the initial capture.

Decomposition of the population of requirements that changed during the project according to the *type* of change is provided in Figure 3.8. The analysis of Figure 3.8 shows a first stage dominated by modification of existing requirements following the initial requirements capture that occurred at PRD1. Although a significant amount of additions occurred at PRD2 (around 36%), requirements modification was the dominant type of change both at PRD2 and PRD3, which were released during the preliminary design stage. Modifications were found in more than 50% of changes in both cases (Figure 3.8). Between the 3rd and 4th releases - a period which included the detailed design stage - Figure 3.7 shows a slight decrease in the number of active system requirements. Figure 3.8 explains the reason behind this decrease, since the proportion of removals ( $\approx 38\%$ ) is higher than the proportion of additions ( $\approx 31\%$ ). Recalling that PRD4 was released shortly after the Critical Design Review (CDR) and before the First Parts (FP) arrived (Figure 3.7), it is also interesting to observe that the proportion of modifications reduced between the third and 4th releases and it is similar to the other types of change (around

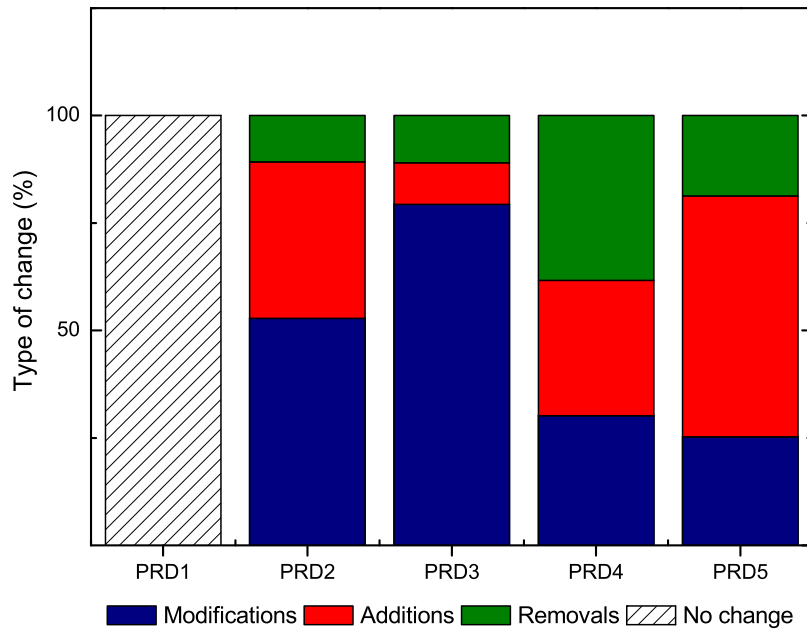


Figure 3.8: Evolution of the types of change throughout the course of the project. Figures correspond to the accumulated proportion of requirements modification, addition and removal during the time period between two consecutive releases of the Product Requirements Document.

30%).

Figure 3.8 also demonstrates that the period comprised between the 4th and 5th releases was largely governed by additions (56% of the total amount of changes at PRD5), which explains the increase in the number of active requirements in the later stages of the project that is observed in Figure 3.7. Recalling the project's time frame, most of the physical testing and prototype development stage occurred during this time period. This effect was partly counter balanced by removals, which accounted for about 19% of changes as shown in Figure 3.8. The requirements evolution found and the type of requirements change found during the time span of the project led to the hypothesis that this requirements engineering process could be divided into three distinct phases:

- Phase I - which started after the initial requirements capture and ended at PRD3 - was mainly governed by modification of existing requirements;
- Phase II - which started after PRD3 release and ended at PRD4 release - was governed by a similar amount of additions, modifications and removals;

- Phase III - which started after PRD4 release and ended at PRD5 release - was governed largely by addition of requirements;

This distinction is supported in the observation that the initial and the late stages were substantially different and in the assumption that the second phase corresponds to a transition between both stages. An initial hypothesis was also that the population of system requirements changes would also have very different root causes of change between the initial and later stages of the process. This hypothesis defined the scope for the subsequent detailed study of causes of change.

It should be noted that these findings - evolution of active requirements and types of change - relate only to *one layer* of the requirements landscape (to system requirements in this case). This layer may be influenced by events occurring in other upstream or downstream layers. A typical example are related events occurring at the environmental layer that result from interactions with the customer, market, certification authorities, etc.

Since these findings characterize the requirements evolution found for the whole gas turbine system, they complement and significantly extend - in a quantitative manner - prior work performed within Rolls-Royce by Pickard et al. (2010) and Nolan et al. (2011), which reported high levels of requirements uncertainty during control systems' development and large amounts of requirements change between the CDR and Entry into Service of the product.

### 3.4.2 Causes of change

Due to the previous hypothesis, the author focused this in-depth study of the causes of requirements change into phases I and III only. Samples representative of the population of changes observed at PRD3 and PRD5 were designed according to the principle illustrated in Figure 3.6. The sample size was computed according to Equation 4.1 for a confidence level of 95% and a margin of error lower or equal than 10%. This resulted in representative samples containing 61 and 122 changes at PRD3 and PRD5 respectively. The assessment of the requirements managers then quantified the causes of changes rela-

tively to previous releases, i.e., PRD2 and PRD4, respectively. Analysis of the statistical results presented throughout this section is based on three main statistical quantities: the estimator of the mean proportion in the population for each cause of change; the estimator of the proportion inside defined strata for each cause of change; and the standard error of the estimator of the mean proportion. Results follow sequentially for each of the two phases previously identified.

### 3.4.2.1 Modification-dominated phase

Statistical analysis showed that the most frequent cause of requirements change between PRD2 and PRD3 was incomplete capture. It was found in approximately 23% of the changes, as presented in Figure 3.9. It was also observed that the requirements content - targets, descriptive information, etc - was progressively added into previously captured versions of the requirements during this early development phase. A typical example of a modification due to incomplete capture is provided in Table 3.2.

Change due to the modification of traceability information or references was the second most frequent cause, found in 18% of the changes. These related essentially to modification of the system requirements' dependency information relatively to requirements captured in higher levels of the landscape, namely to business and customer requirements. An example is also included in Table 3.2. Changes driven by the customer were the reason for approximately 15% of the total number of changes, representing the third most frequent cause (Figure 3.9).

Incorrect or ambiguous language and incomplete flow down appear as the fourth and fifth most frequent, respectively, with mean proportions around 10% (Figure 3.9). Many examples of requirements modifications were found during this time period arising from semantic issues. An example that can be provided here is shown in Table 3.2, where the expression *"shall not hazard the super-system"* was modified subsequently to *"shall not result in structural failure or hazardous system effects"* in a particular system requirement due to incorrectness or ambiguity in the language previously employed.

All together, the top five causes accounted for more than 75% of changes. Various other



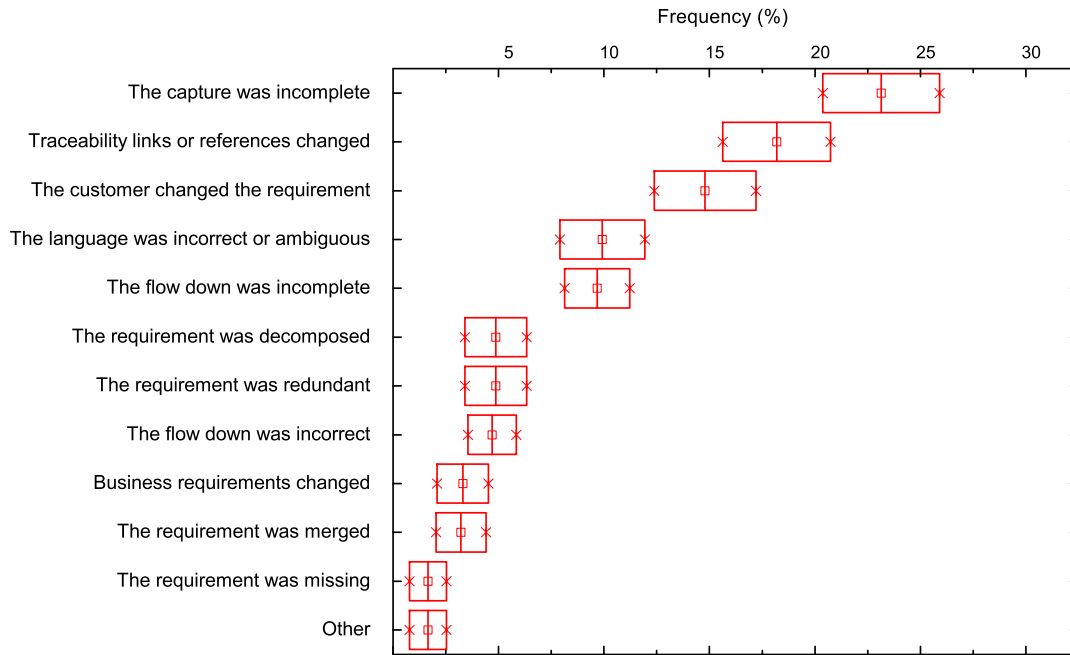


Figure 3.9: Estimator of the mean proportion of each cause of change during the process phase dominated by modifications, between PRD2 and PRD3. A 50% confidence interval of the mean proportion in the population is displayed.

causes - business changes, missing requirements, incorrect flow down, redundancy, decomposition and merger - account for the remaining proportion, with individual mean values below 5% (Figure 3.9).

Decomposition of the data according to the type of change provides additional insight, which is presented in Figure 3.10. The results demonstrate that all additions performed between PRD2 and PRD3 were caused by incomplete flow down of requirements. These relate to further decomposition of requirements from the environment layer - customer, business or certification requirements - down to the system level. Moreover, Figure 3.10 also shows that incorrect flow down was the most frequent cause of removals during this period. This results from awareness that a particular requirement should be captured in or cascaded to another level of the landscape.

Change caused by the customer, requirements redundancy, decomposition and merger were also responsible for part of the removals (Figure 3.10). Within the modification stratum, the most frequent causes of requirements change were identical to the ones found for the whole population. This result is expectable, recalling that during this time

Table 3.2: Examples of changes in requirements found between PRD2 and PRD3. The text underlined relates to the changes observed.

Version at PRD2	Version at PRD3	Cause of change
The system shall be capable of starting at X between Y°C and Z°C with special starting procedures to raise the oil temperature. {Trace to: BRD - ...}	The system shall be capable of starting at X between Y°C <u>(the hard limit of the oil temperature)</u> and <u>W°C (the lower limit of the start environmental envelope as defined in Ref. H)</u> with special starting procedures to raise the oil temperature. {Trace to: BRD - ...}	Incomplete capture
Fires shall not hazard the super-system. {Trace to: BRD - ...}	Fires shall not <u>result in structural failure or hazardous system effects.</u> {Trace to: BRD - ...}	Incorrect or ambiguous language
A strategy, plan and governance structure shall be defined to ensure that the system achieves a measured level of product maturity pre-agreed by the customer by the first customer system FPS date.	A strategy, plan and governance structure shall be defined to ensure that the system achieves a measured level of product maturity pre-agreed by the customer by the first customer system FPS date. <u>{Trace to: BRD - ...}</u>	Traceability links or references changed

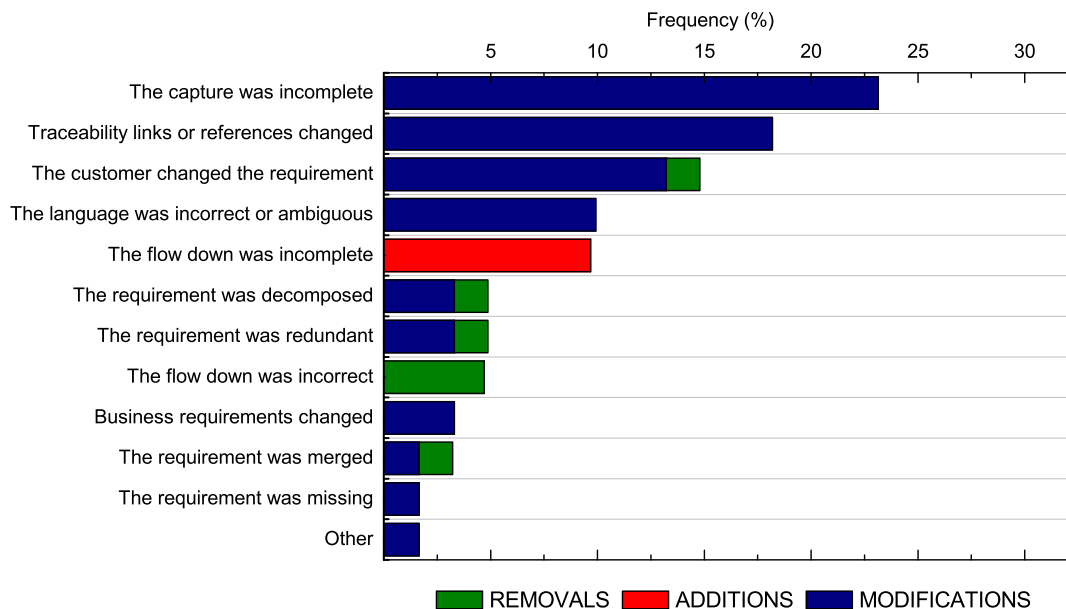


Figure 3.10: Estimator of the mean proportion in the population according to the type of change, between PRD2 and PRD3.

period modifications were the dominant type of change.

#### **3.4.2.2 Addition-dominated phase**

This period was noticeably different from the previous. Results support that incomplete flow down was in this case the most frequent cause of requirements change. Figure 3.11 shows it was found in 34% of the changes. Strata results presented in Figure 3.12 demonstrated that more than 60% of additions occurred due to incomplete flow down.

The analysis revealed that requirements already existed in the environment layer, but they had not been formally cascaded and re-engineered at downstream levels. The author observed in strata results that the flow down was particularly intense into some types of requirements, such as into performance, structural, operational and life requirements. Empirical results supporting this observation are presented in Figure 3.13. Performance relates to many functionalities that the gas turbine system must deliver at different operating conditions, structural requirements specify the level of integrity the system must contain at all times and life requirements specify the cyclic capabilities of the system under different operational conditions. It was also observed in the data that the flow down of certification requirements was also responsible for a proportion of the additions.

Change due to new customer requirements was found in almost 15% of the cases (Figure 3.11), becoming the second most frequent. The customer induced about 19% of additions, approximately 17% of removals and a relatively small proportion of modifications, about 3% (Figure 3.12). An interesting finding from strata results relates to the distribution of customer driven change. Figure 3.13 shows it was more intense in performance requirements, which could be expectable considering that the gas turbine must deliver a high number of functionalities which have associated performance targets. It demonstrates also that the remaining proportion of change generated by the customer was similarly distributed throughout a variety of other types (program, mass, life, reliability and environmental requirements). The fact that a similar amount of change driven by the customer - about 15% - was found both in the early and in the late stage of the process is a relevant finding.

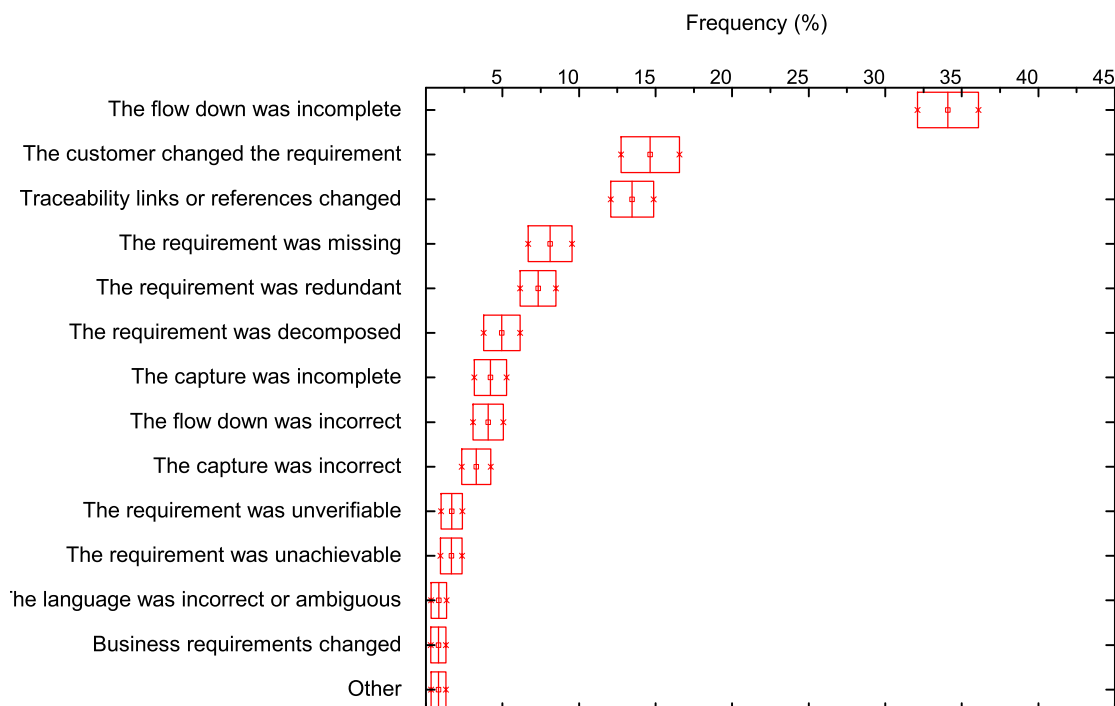


Figure 3.11: Estimator of the mean proportion of each cause of change during the process phase dominated by additions, between PRD4 and PRD5. A 50% confidence interval of the mean proportion in the population is displayed.

Customer induced change was followed by traceability generated changes, which accounted for approximately 13% of the total amount of change accumulated between PRD4 and PRD5 (Figure 3.11). These were the main bulk of modifications, as seen in Figure 3.12. Figure 3.11 also shows that missing and redundant requirements became the fourth and fifth most frequent causes during this time period, respectively with 8% and 7%. Further strata analysis demonstrated that missing requirements appeared as additions (Figure 3.12) and were concentrated in only a few types of requirements, such as in transportability or structural requirements. In the former, the time of this capture relates to increased understanding of transportability issues at this stage arising from the testing program of physical prototypes.

Redundancy was the most frequent cause of removals (about 39%), as seen in Figure 3.12. A deeper analysis revealed that in fact many removals due to redundancy were a direct result of addition of new requirements due to incomplete flow down. A *knock-on effect* was thus established between both types of change during this requirements engi-

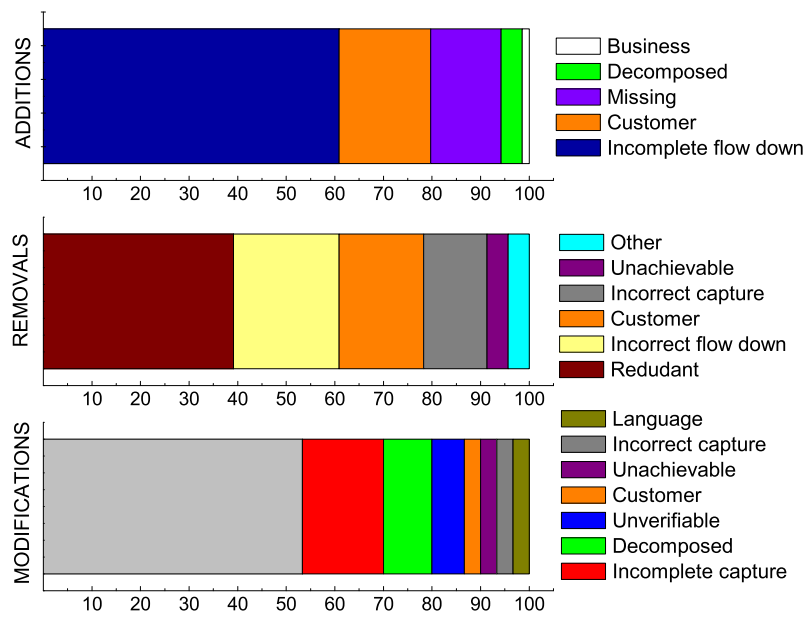


Figure 3.12: Estimator of the proportion of each cause of change inside strata defined according to the type of change, between PRD4 and PRD5.

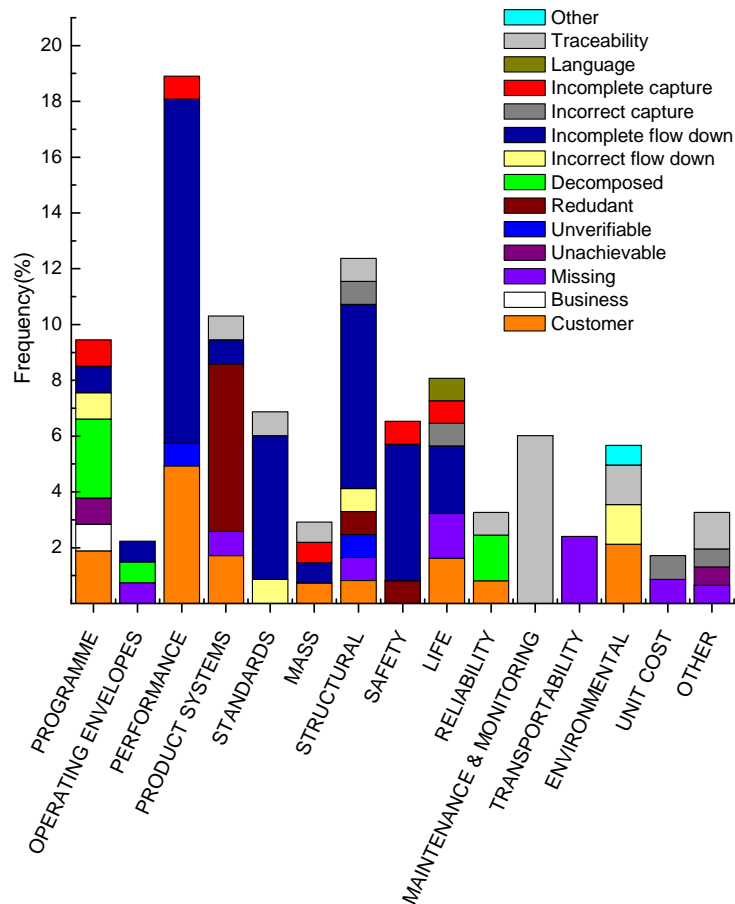


Figure 3.13: Estimator of the proportion of each cause of change inside strata according to the type of requirement, between PRD4 and PRD5.

neering phase. Strata analysis also revealed that the majority of redundancies occurred in requirements previously captured for individual gas turbine product systems - e.g. oil systems, fuel systems, etc.

These five causes led to more than 77% of the total number of changes between PRD4 and PRD5. Decomposition was found in about 5% of the causes and incomplete capture and incorrect flow down were each found in about 4%. Incorrect capture, unachievable and unverifiable requirements and business changes complete the set of causes found during this period with decreasing proportions (Figure 3.11). It is interesting to notice that this small proportion of validation and verification related changes did not appear during the early process phase. Analysis of Figure 3.13 showed that they occurred essentially in program, performance and structural requirements.

### **3.4.3 Inter-layer analysis**

Considering the previous results, the author proceeded with an inter-layer analysis of the requirements engineering process performed during the development of this new gas turbine at Rolls-Royce. The aim was to further understand the root causes of change, namely the causes arising from interdependencies between system requirements and requirements captured in other layers of the requirements landscape.

These interdependencies were visible in the amount of change triggered by incomplete flow down, for instance. The main findings generated from this inter-layer analysis are represented in Figure 3.14. This figure joins the project's time frame and the main events that occurred at each layer of the landscape which is upstream of the Product Requirements Document. It includes also the *Environment* layer - containing requirements captured from the customer and from certification authorities - and the *Enterprise* layer, which contained requirements captured from internal business related functions.

It was found that, at the time of initial capture, PRD1 captured customer requirements from the MoU (issue 2) and business requirements from issue 1 of the Business Requirements Document (BRD) and engineered them into system requirements. The timing of the flow down process is illustrated in Figure 3.14. During the modification-dominated

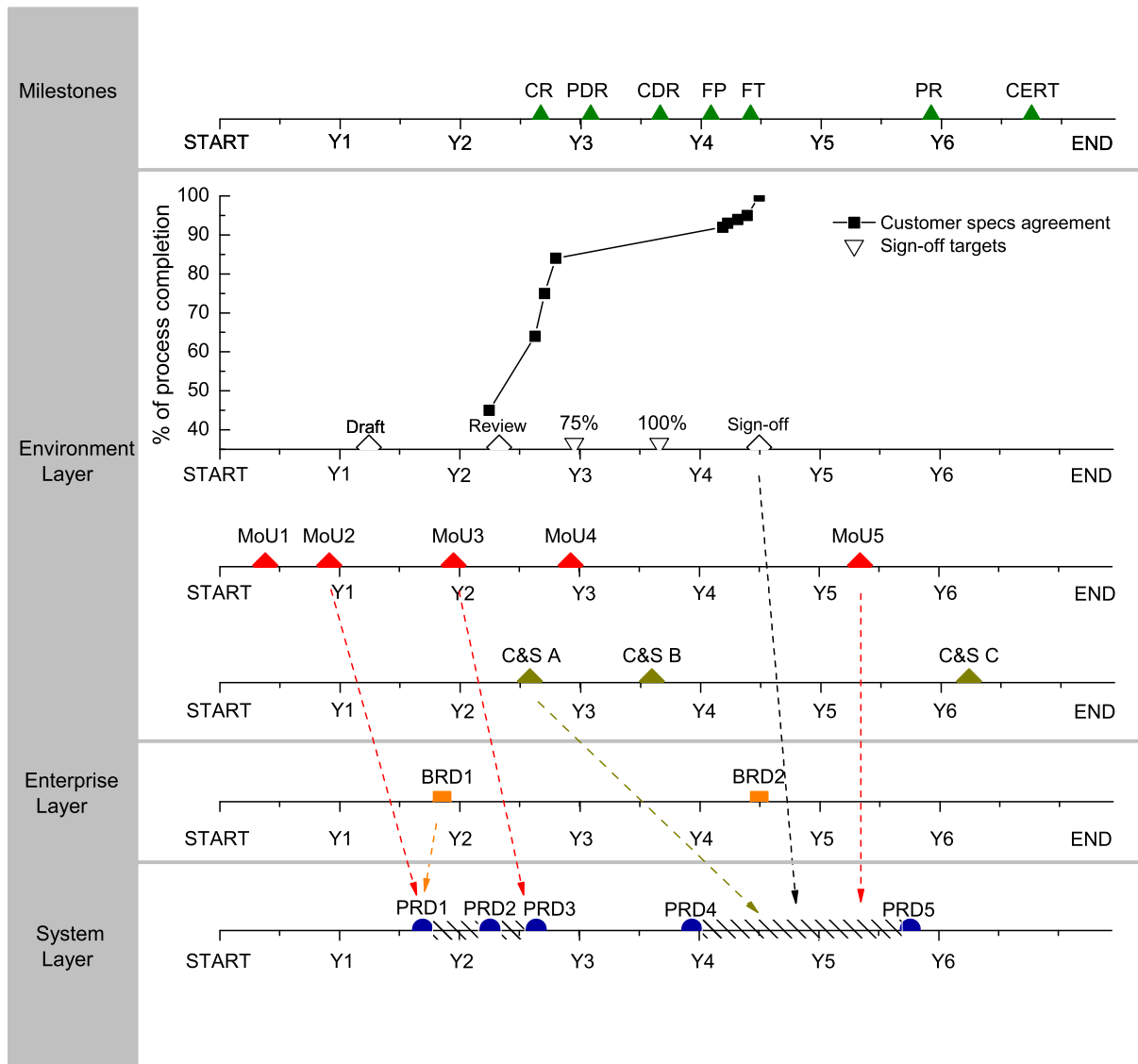


Figure 3.14: Inter-layer analysis of the requirements engineering process during the development of a new gas turbine. Arrows represent flow down. Legend: C&S - Certification and Standards.

phase of the project, part of the changes were triggered by the release of issue 3 of the MoU. This document contained new customer requirements and updated critical targets - such as constraints in maximum allowed weight or fuel consumption - that were incorporated into PRD3 in a response to changing customer needs. This interaction is the origin of 15% of the changes previously quantified between PRD2 and PRD3.

In parallel to the signature of the various MoUs, the author found that a continuous interaction process was going on with the customer. This interaction consisted in the review and agreement process of the main bulk of customer specifications for the new gas turbine, consisting of *thousands* of technical specifications about the system.

Archival research demonstrated that the initial draft of the customer's technical specification documents was finalized at the end of the project's first year and that a review and negotiation process started between appointed customer and Rolls-Royce technical specialists. Data collected suggests that almost 50% of the specifications were already agreed when the initial review finished. Moreover, as shown in Figure 3.14, the agreement status increased rapidly and by the end of the second year about 75% of the total amount was agreed (Figure 3.14). This period coincides with the period comprised between PRD2 and PRD3 at system level.

Considering the sign-off targets that had been specified by the project, the agreement was on track at that time as illustrated in Figure 3.14. However, the agreement rate slowed down during the third year. The process was at 90% status at the targeted signature date and the data collected showed that nine months more were needed to complete the agreement process (Figure 3.14). The official sign-off occurred during the second quarter of the fourth year.

Capture of the customer specifications occurred in the requirements database shortly after and the author's investigation found that the systematic flow down and re-engineering process of these specifications into system requirements started subsequently. This event explains a large share of additions due to incomplete flow down found during the period comprised between PRD4 and PRD5. It also means that requirements were *available earlier* but were only captured and engineered into downstream layers after the official signature.

Furthermore, the author found also that certification specifications had been captured in the requirements management software tool during the project's second year (Figure 3.14), but only begun to be flown down to the system level after the fourth year. This second process also explains part of the new requirements found at PRD5 release.

Finally, a new MoU version (issue 5) was also signed late in the project which contained additional changes in customer needs. This new set was largely responsible for the amount of change caused by the customer (around 15%) that was quantified during the previous study.



### **3.4.4 Discussion**

Requirements change in large technical systems is challenging for management. The previous results demonstrate that there are many root causes of change and its proportion may vary considerably during the time frame of a complex development project. Because of that, a thorough discussion of the findings is required.

An aggregated analysis of the results from the case-study shows that the vast majority of requirements change – around 80% – is self-induced both during early and late product development stages. A wide range of causes was found contributing to internally-driven changes of the requirements that had been previously released: incomplete capture, incomplete flow down, incorrect or ambiguous language, missing requirements, redundancy, decomposition, etc (Figures 3.9 and 3.11). Conversely, it was observed from the case-study findings that externally-driven change – arising from uncertainties typically interpreted as outside the organization’s control – accounted for a smaller part of the total amount of change (15 to 20%). Among all the high-level stakeholders, the study demonstrated also that externally-driven change derived essentially from the customer.

An interesting question arising from these findings is the relationship between each cause of requirements change and the impact of the corresponding implementation of change on the development process or on program costs. In this regard, empirical observation suggests that changes in requirements triggered by decomposition, redundancy, merger, lack of traceability or validation and verification issues have low impact/cost since their implementation implies essentially carrying out the activity of re-engineering those requirements in the database.

Conversely, changes due to incomplete or incorrect flow down and incorrect or ambiguous language are deemed as having a higher impact/ cost since its implementation may imply the rework of components or sub-systems affected by the requirements suffering such changes. Empirical knowledge also suggests that the level of rework on sub-systems and components may be much higher when change arises from the customer and business functions, when it is due to incomplete or incorrect elicitation or when it is triggered by missing requirements. These root causes of change may lead to major engineering changes

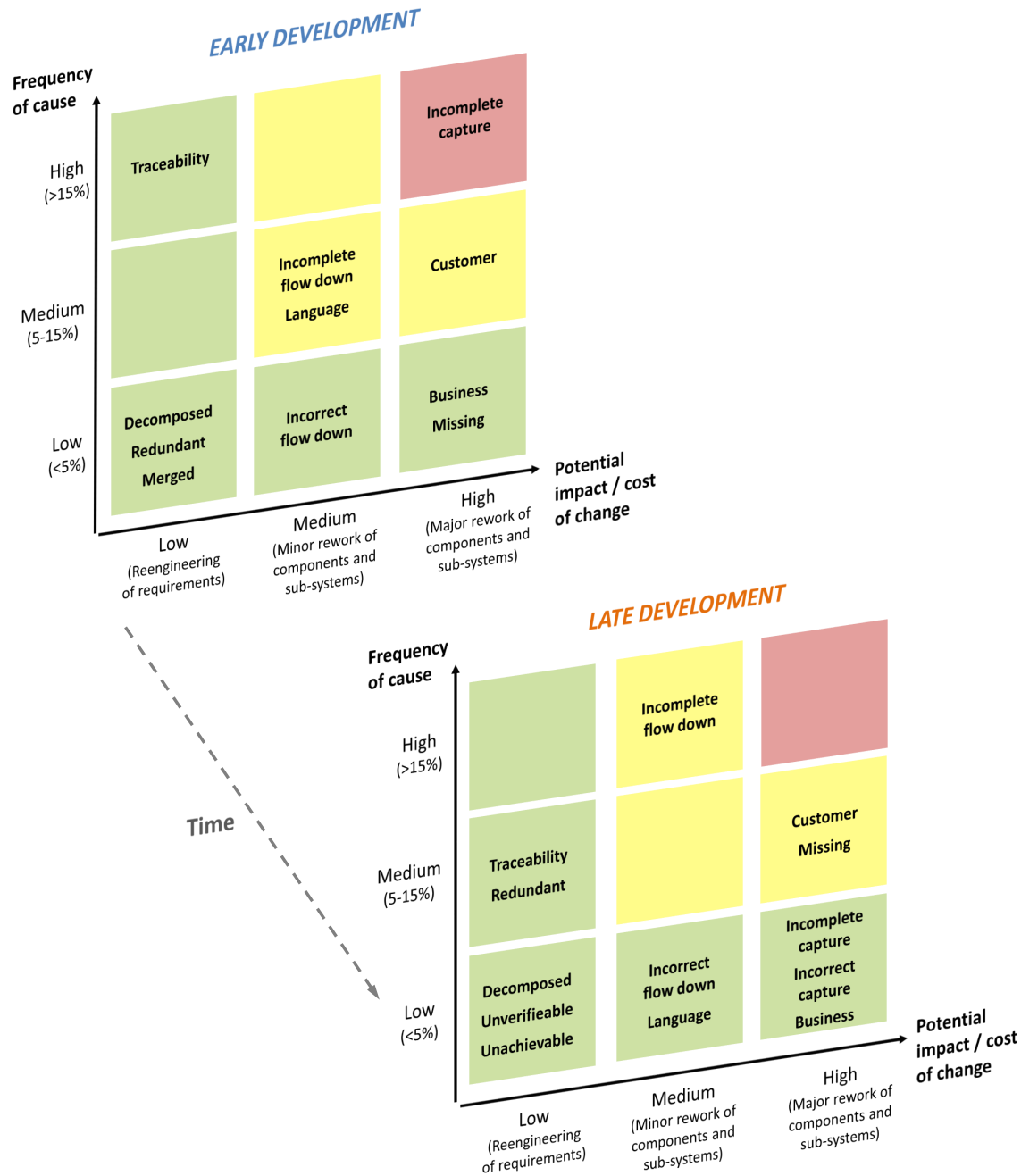


Figure 3.15: Frequency of causes of change from case-study findings versus potential impact/ cost of change.

on the design solution itself or to the late discovery of hardware problems during prototype development and testing, which normally have significant impact on both product development time and program costs.

Figure 3.15 presents the frequency of causes of change from the case-study findings against the potential impact/ cost of each change implementation. The latter is evaluated qualitatively according to the author's previous arguments derived from empirical observation

acquired during the study. Figure 3.15 illustrates that during early development most of the rework affecting components and sub-systems may be originated due to incomplete capture and change generated by the customer, which occurred with medium to high frequency. Missing requirements and customer-induced change were changes with potentially high impact/cost of change during late development stages and also with a medium frequency of occurrence. At the opposite end, various sources of change associated with low impact/ cost of implementation occurred during both early and late development stages with low, medium and high frequencies (Figure 3.15).

Recalling Figures 3.9 and 3.11 – which quantified the mean proportions of each cause during both stages – the causes of change related to high implementation costs account for approximately 40% of the total amount. This proportion includes both internally and externally-induced change and there is no dominance of either kind. A major hypothesis that emerges from these findings is that most of the cost incurred by organizations due to changes in requirements results from a relatively small proportion of the amount of change that is actually performed during complex development projects.

### **3.5 Improvement guidelines for management**

From a manager’s point of view, there are normally good reasons for desiring to reduce the amount and/or the impact of requirements change on development time or program costs. The research methodology presented in section 3.3 led the author to conduct group sessions with the engineers responsible for managing system requirements during the development of the new gas turbine. The previous case-study findings were analyzed jointly. Several practical management guidelines aiming to further improve the requirements engineering process emerged from these sessions, with the goal of reducing the amount or the impact of requirements change. The following sections present and discuss them.

### 3.5.1 Front-load the requirements engineering process

The author's investigation observed that the flow down process of many technical specifications into system requirements aiming to support design and development activities only occurred late in the process. Several reasons may justify this fact. Firstly, complex technical systems, such as the one developed by Rolls-Royce, can have *very long* negotiation processes over customer specifications. Organizational policies may dictate that requirements should only be flown down when the process is complete to avoid wasting engineering resources. In such cases, the organization is *knowledgeable* of what the product needs to deliver but chooses only to engineer the requirements when it is sure of its content. Constraints over human resources available to perform the requirements engineering job may also justify the postponement.

However, the outcome of the group discussion between the requirements managers and the author at Rolls-Royce favored *front-loading* the process to reduce the impact of late changes and the risk of proceeding with extensive testing programs without having all requirements engineered and in place. The cost of hardware problems in such cases was perceived to be *much higher* than the cost of wasted engineering resources during earlier product development stages due to potentially larger amounts of change. The aim of this strategy is thus to reduce the *impact* of requirements change through a reduction in the amount of change occurring late, but not necessarily through a reduction in the total amount of change over the project's lifespan.

The effects of front-loading the requirements engineering process are represented in Figure 3.16. The process envisioned is one capable of *stabilizing* the number of customer requirements captured and the number of system, sub-system and component requirements engineered before the design freeze, which is planned to occur at the Critical Design Review.

### 3.5.2 Increase the level of concurrent requirements engineering

Front-loading the requirements engineering process demands a great deal of concurrent requirements engineering. It came out from the analysis of the statistical results by the

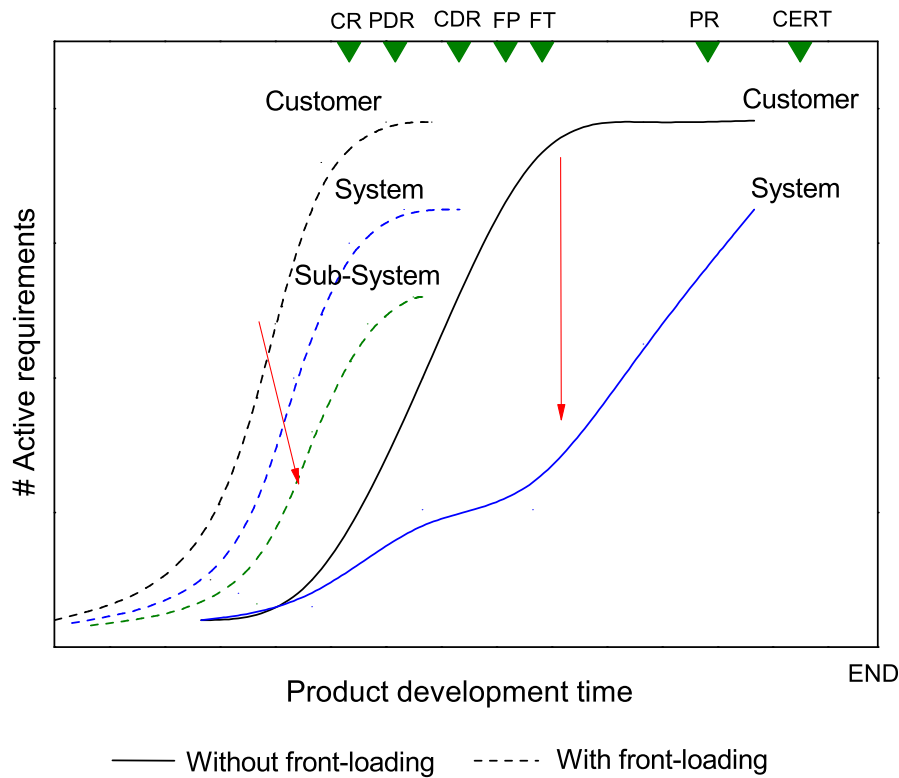


Figure 3.16: Simplified representation of the effects of front-loading the requirements engineering process. Arrows represent flow down.

requirements managers that the flow down process of requirements that have been *reviewed and agreed* can occur in parallel to the negotiation and agreement process with the customer. This principle of concurrency can be extended downstream across all layers of the landscape. Concurrency in requirements engineering is illustrated in Figure 3.16 by the small delay in the stabilization of the number of active requirements across different levels of the landscape.

The group of requirements managers discussed the level of coordination and integration between requirements engineering teams that is needed in organizations such as Rolls-Royce to implement this practice. Concurrent processes must be anchored in *robust* change management processes which standardize how day-to-day procedures must be performed. These procedures should specify how a team responsible for an higher level requirement - e.g. a system requirement - should engage with a team responsible for a dependent lower level requirement - e.g. a sub-system requirement - to communicate, evaluate, sign-off and implement a requirements change. Monitoring change implementation across system levels demands also a joined structure between the requirements database

and the solution definition documentation, which points to additional developments in the IT infrastructure in place in the organization. The standardization of procedures and database integration goes beyond the traditional change management processes found in literature and previously discussed in Section 2.3.

Furthermore, an increase in the level of process concurrency demands also an increase in the frequency of regular coordination meetings between the teams involved. These add organizational complexity and costs, but an increase in the level of requirements engineering process concurrency was perceived as an improvement guideline for management.

### **3.5.3 Allocate more resources earlier**

Under normal market conditions, organizations developing complex technical systems typically struggle to find enough engineering resources to perform the design and development activities which are needed. However, concurrent processes are certainly difficult to implement without the right amount of resources allocated to requirements engineering.

In complex technical systems containing thousands of higher level specifications, the process of flowing down and re-engineering these specifications involves a *huge* amount of background engineering work to ensure that the new requirements are complete, independent, atomic, unambiguous, etc. It is known from practice that under-resourced teams will tend to serialize requirements engineering activities. Because of that, serializing the process increases the chance that the flow down process of many requirements into system, sub-system and component requirements occurs late in the process.

Allocating more resources is thus a management guideline that follows from the idea of increasing concurrency in the requirements engineering process. In addition, front-loading the process demands that resource allocation ramps-up during the preliminary design stage at all layers of the landscape, so that the capture and flow down process is complete before the Critical Design Review. It thus means that more resources need to be allocated to requirements engineering activities *earlier* in the process.

### 3.5.4 Invest in practical training

The previous strategies search for a reduction in the impact of late change due to its known effects on development time and program costs. In addition, management guidelines aiming to reduce the *amount* of requirements change also emerged from the group analysis of the causes of change reported earlier in this Chapter.

The statistical results demonstrated that a significant proportion of change arises from incorrect capture or flow down, ambiguous language, issues with traceability, redundancy, etc. Discussion of many examples with the requirements managers pointed out that the capture of a single requirement is often a quite challenging process since it involves several areas of expertise that engage into negotiation about its content, language, dependencies, etc. This process is essentially *iterative*, which helps to explain the previous types of change. Because of that, a lot of "learning while doing" occurs.

Since the process is highly iterative and practical, the capability of mastering the requirements engineering and management process and reducing the amount of internally-generated change depends on personal experience and skills which are typically accumulated *over time*. This knowledge may be lost when people move jobs in the organization, a situation which occurs often in large firms such as Rolls-Royce. A management guideline that came out from this discussion was the need to invest in *practical* requirements engineering training in organizations, in opposition to the traditional training model which tends to remain at a high level - explaining the general principles, the main stakeholders, the document landscape, the deployment process, etc. Conducting practical training - one relying heavily on example-based learning - across the organization was a guideline recommended also to ensure that requirements management knowledge is preserved and consequently future projects are able to reduce the amount of self-generated change.

### 3.5.5 Set-up validation and verification systems

It was also found that a part of the requirements was discovered to be unachievable or unverifiable during later stages of the process. The capture process can often occur without enough time being spent on understanding and documenting how particular requirements

should be validated and verified. This typically occurs due to time and budget pressures that organizations face. However, organizations such as Rolls-Royce recognize that validation and verification efforts promote higher requirements quality and stability in the longer-term.

Setting-up validation and verification systems was thus pointed out as a guideline to address this cause of requirements change. This system can be designed to best fit the organizational needs of each firm. Developing and using computational algorithms that test if the requirement is complete, semantically correct and traceable is one possible approach to perform requirements validation. Introducing a standardized template attached to each requirement that allows the engineer to document how the requirement will be satisfied (through inspection, computational analysis and simulation, physical testing, etc) is also one possible avenue to conduct requirements verification. The system in place should also allow metrics calculation so that management can inspect the amount of requirements that have been specified with validation and verification checks in product development projects.

### **3.5.6 Effects of the guidelines in the process**

Front-loading, increasing the level of requirements engineering concurrency, allocating more resources earlier, investing in practical training and setting-up validation and verification systems were various management guidelines identified and discussed. The effects envisioned on the requirements engineering process are illustrated in Figure 3.17. This qualitative representation of the requirements change evolution aims to point out that there are two main improvements targeted by these guidelines. The first is a reduction in late requirements change. And the second is a reduction in the total amount of change observed during product development projects.



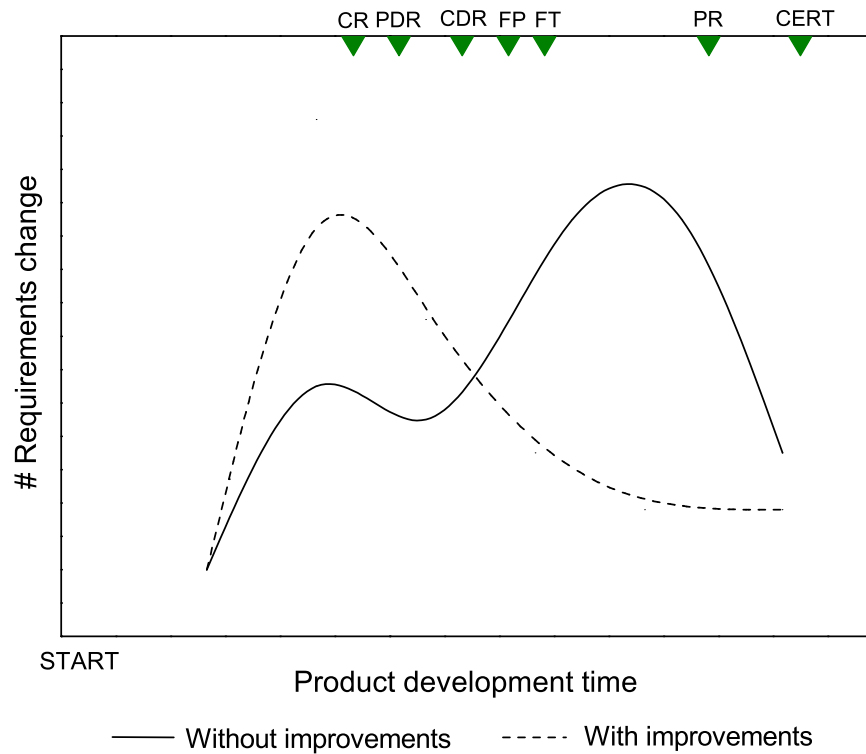


Figure 3.17: Simplified representation of the effects of the management guidelines captured in the case-study on the evolution of requirements change during the product development process.

## 3.6 Review of progresses

The chapter has presented and discussed findings from a large-scale empirical study to the root causes of requirements change during complex product development. The case-study research approach implemented by the author reconstructed quantitatively and qualitatively how system requirements evolved and changed during the course of the development of a gas turbine. Looking back to the research questions presented in Section 1.4 which motivated this investigation, the author argues the findings described in this Chapter answer them in the following manner:

**Q1** *How do requirements evolve and change during complex product development projects, such as the ones executed by Rolls-Royce?*

The data collected during the empirical study showed that the number of active system requirements roughly doubled during the course of the development process, when compared with the number at the time of initial elicitation. The evolution revealed also that the

stabilization of the number of system requirements may occur late in the project, typically during prototype development and testing. Furthermore, the investigation demonstrated that system requirements evolve through modifications, additions and removals and that the early development stages were dominated by modification of existing requirements following the initial capture, while late development stages were governed mainly by additions. In addition, it was found from the research that long requirements negotiation and agreement processes with high-level stakeholders, such as the customer, greatly impact the evolution of system requirements during complex product development.

**Q1-b** *What are the root causes of requirements change during the design of large technical systems?*

The methodology followed during the study led to the identification of a wide range of causes of requirements change that typically occur during the development of large technical systems. Change arising from high-level stakeholders – such as the customer, certification authorities or business functions – was one of the key causes. Issues surrounding requirements elicitation were found as a major causes of change, namely missing requirements, incorrect or incomplete capture, ambiguous language, redundancy and decomposition or merger of requirements that had been already captured. Incorrect or incomplete flow down appeared as a source of change related with problems in the requirements cascading process. In addition, management of traceability information across the document landscape was also found to be a source of change during projects.

**Q1-c** *How do the causes of change vary during different phases of the product development process?*

The study showed that the early development stages were dominated by requirements modifications due to incomplete elicitation of requirements that had been previously captured. Traceability issues, customer driven change and incorrect or ambiguous language were also relevant during this stage. Conversely, the author found that late development stages - namely during prototype development and testing - were governed by requirements addition due to incomplete flow down of high-level technical specifications.

Customer induced change, missing requirements, traceability and redundancy were also frequent during this phase. The five most frequent causes of change encompassed more than 75% of the total amount of change during both early and late stages of the project. Knock-on effects between some of the causes of change were also observed.

**Q1-d** *How much change is externally and internally driven?*

The question concerning the proportion of externally and internally-driven change has been clearly answered. The empirical study found that more than 80% of changes generated during the development of the complex system studied in this Chapter had internal root causes. It was also observed that the amount of change generated from the customer remained relatively constant during the development process, around 15%, which is interesting finding. The author's belief is that this trend is general, but the exact proportion of external versus internal change may vary across projects and industries.

## 3.7 Summary

In summary, this chapter contributes to an increased understanding of the requirements evolution and the causes of change found at different phases of the product development process of complex technical systems. The author presented findings from a large-scale empirical-study performed at Rolls-Royce which investigated a new gas turbine developed by this manufacturer during the course of 6 years. Using both quantitative and qualitative research methods, the author collected and studied the requirements database of this system. The database contained 700 system requirements and its evolution was embodied in more than 1000 changes, which occurred during the product development process. The research approach generated original results about requirements change during complex product development process which extend current knowledge on the topic and prior research contributions reviewed in Chapter 2.

The issue of result generalization always arises from case-study approaches (Yin, 1984). However, the author argues that the main findings contained in this dissertation are generally representative of complex product development processes. It is common that large

technical systems require long development processes which make them more vulnerable to external uncertainties and long and laborious negotiation processes with multiple stakeholders about high level technical specifications. Requirements management processes are thus highly challenging for organizations developing such systems. The case-study showed that most requirements change is self-induced due to a variety of different causes, being incomplete capture and incomplete flow down some of the most frequent during early and late development stages, respectively. Considering that Rolls-Royce is one of the most competitive companies in the world, this thesis argues that the same finding is likely to be encountered in many organizations developing complex technical systems.

Furthermore, the findings described and discussed in this chapter were able to answer the research questions – Q1, Q1-b, Q1-c and Q1-d – that had been established in the initial research plan. The chapter’s key conclusions retrieved from these findings are:

- Requirements evolve and change in organizations developing complex technical systems through modifications, additions and removals and the number of requirements at a particular system layer may double during the course of the development process and only stabilize late in the project, already during prototype development and testing.
- The root causes of change vary during different phases of the development. Empirical data shows that early development was governed by modifications due to incomplete elicitation, while late development was dominated by incomplete flow down of higher-level specifications.
- More than 80% of changes generated during the development of the complex system studied had internal root causes and the amount of change generated from the customer remained relatively constant during the development process, around 15%.
- Empirical findings suggest that the causes of requirements change with potential high impact on development time and program costs account for a relatively small proportion – around 40% – of the total amount of change that is actually performed during complex development projects.

The conclusions stated above constitute major contributions from this empirical research. This knowledge constitutes a general academic contribution from this Chapter, which complements previous empirical studies – e.g. Almfelt et al. (2006); Sudin and Ahmed (2009); Vianello and Ahmed-Kristensen (2012) – and recognized literature – e.g. Kotonya and Sommerville (1998); Robertson and Robertson (2006); Pohl (2010) – on the topic of requirements engineering and management.

In addition, these findings led the group of engineers involved in the research to analyze various management guidelines aiming to improve requirements engineering in practice. Front-loading the requirements engineering process was proposed as a way to reduce the impact of late change. Increasing the level of concurrency in requirements engineering processes was discussed to ensure that the number of requirements captured and engineered stabilizes roughly at the same time and before design freeze at all levels of the landscape. Allocating more engineering resources earlier in the process was a management guideline that followed from the previous two. Moreover, investing in example-based training across the organization and setting-up validation and verification systems were recommended to ensure that future projects are able to reduce the amount of internally-generated change. The previous management guidelines ultimately aim to attain processes that generate less changes and changes with lower potential impact/cost since they occur earlier in the development process. This thesis argues that the management guidelines captured in the study are also useful for other organizations performing product development and practitioners in systems engineering and requirements management.

In conclusion, empirical research performed by the author at Rolls-Royce allowed academia and industry to understand better the requirements evolution and the causes of change, generating unique contributions and novel knowledge. This dissertation proceeds next with the proposal and evaluation of a new method for comprehending and handling uncertainty and change in requirements and other critical design information exchanged between teams during complex collaborative and distributed development processes.



## Part II

# Managing uncertainty

People search for certainty. But  
there is no certainty.

—RICHARD P. FEYNMAN





# Chapter 4

## A method for imprecision management

Exploratory research described in Chapter 1 revealed that lack of knowledge about the level of uncertainty in design variables – such as requirements – which are communicated periodically between design teams during complex product development generates many practical management issues. The interviews performed by the author showed that questions such as "shall I wait for more accurate requirements?", "how much change should I expect at this development stage?", "if I begin now with this requirement, what is the risk of having to do it all over again?" or "will I get any time saving later if I begin to work with uncertain inputs?" arise often in practice.

The author argued in Section 1.2 these questions reflect the engineers' need of comprehending and handling uncertainty in the design information exchanged during collaborative and distributed engineering processes. This chapter presents and evaluates a method synthesized by the author to support engineers and managers taking such process management decisions with information about the typical level of uncertainty that can be expected in requirements or other design variables which are inputs to design activities performed by different development teams<sup>2</sup>. The method was synthesized from the archival research and empirical observations that the author was allowed to perform during his eight months

---

<sup>2</sup>This chapter is largely based in the material previously presented in Fernandes et al. (2014a) and Fernandes et al. (2014b). Reprint is made with permission.

stay at Rolls-Royce Derby.

Discussion proceeds in six main steps. Firstly, the research objectives are introduced. Secondly, the different types of uncertainty found in engineering design literature are reviewed and research gaps are pointed out and discussed, since these have inspired the approach followed by the author. Thirdly, the method aiming to support organizations understanding, quantifying and communicating uncertainty in requirements or other key design variables is presented. A case-study performed at Rolls-Royce testing the method's applicability in practice is then discussed, followed by a critical analysis of its strengths and limitations. The chapter ends with a discussion of research progresses and a summary of conclusions.

## **4.1 Objectives**

Understanding and managing uncertainty levels in requirements and other key inputs is important for organizations carrying out complex product development processes. The research setting found at Rolls-Royce and described in Section 3.2 shows that there is a co-evolution of both requirements and design solutions at different layers of the system's hierarchy during time. Moreover, the systems engineering approach used in the development process drives the cascade of requirements from higher system levels to lower levels *both* from parent requirements existing in the layers above *and* from the solution definition generated also in higher levels. This principle was represented in Figure 3.2 and the flow down from both parent requirements and solution definition parameters at Rolls-Royce has been pointed out in Figure 3.3.

A consequence of this approach is thus the existence of interdependencies linking design activities performed by different teams which are responsible by the synthesis of solutions for different parts of the system. These interdependencies imply that changes in one particular design variable may affect and impact the outcome of activities performed at another system layer, which is dependent upon the value that has been assigned to it. For instance, teams working concurrently to deliver sub-system design solutions may be impacted by the uncertainty leading to change in the requirements flown down by upstream

levels. Looking into Figure 3.3, the uncertainty surrounding customer or system requirements cascaded to sub-systems or surrounding the sub-system requirements derived from the system's solution definition may lead to changes that generate the difficulties captured by the author during exploratory research (Chapter 1). Another typical situation observed often in practice is the uncertainty in the definition of interface variables which may potentially affect several sub-system and component design teams.

Design variables – such as requirements – communicated by a design team and used as inputs to activities performed by other teams are naturally imprecise. *Design imprecision* is a form of uncertainty coined in literature (Wood et al., 1990; Otto and Antonsson, 1994; Antonsson and Otto, 1995) to express uncertainty in decision-making during the design of a new product. It arises when designers and engineers have not yet acquired sufficient knowledge or possess enough confidence to decide what is the precise value that a design variable should take and it is higher during the early stages of the design process.

The goal of this chapter is to present a method for imprecision management supporting the needs of product development intensive industrial organizations and particularly those engaged in the design of large technical systems, such as Rolls-Royce. The method enables engineers to understand, quantify and communicate the level of imprecision to be expected in requirements or other critical variables used regularly as inputs by different teams during design processes. The author proposes a structured approach to imprecision management relying on five main steps: collection of historical records; reconstruction of the design variables' time evolution; statistical characterization of imprecision; communication of the typical levels of imprecision that can be expected to the participants involved in new product development projects; and knowledge update from new projects.

Furthermore, this chapter aims to present and discuss empirical findings that resulted from a case-study performed by the author at Rolls-Royce with the intent of testing and exemplifying the imprecision management method proposed. The author collected records that allowed to follow 36 design variables across multiple product development projects realized at this manufacturer. These were functional requirements defined by the system design team and communicated regularly to different sub-system teams. Un-

certainty surrounding such variables greatly affects the duration of the collaborative and concurrent design process involving both system and sub-system teams. The empirical study performed by the author revealed quantitative and qualitative interesting findings which this chapter aims to explore.

The author's purpose was to provide an answer to the second set of research questions defined in Section 1.4. The study intended to comprehend how does the level of uncertainty in such design variables varies during product development (Q2-b) and what is the typical level that should be expected by designers engaged in new projects (Q2-c). This chapter will also seek to demonstrate that the answers to these research questions, retrieved from a real-world industrial environment, represent new contributions to engineering design research. The chapter follows with a review and discussion of prior literature on this topic.

## **4.2 Uncertainty in design**

Uncertainty definition, quantification and management is a foundational topic that has concerned researchers across a wide range of sciences and applications. An in-depth survey performed across the social sciences, physical sciences, engineering and management sciences by Thunnissen (2005) showed that definitions and taxonomies vary considerably across these fields.

For instance, Thunnissen (2003) reported that recent economics acknowledges the existence of fundamental uncertainty (situations where insufficient information conducts to unreliable probabilities) and ambiguity (uncertainties about probabilities that could be known), while the management science defines aleatory uncertainty (arising from randomness in a system), epistemic uncertainty (originating from lack of knowledge of a system), parameter uncertainty (surrounding the true values in mathematical models), model uncertainty (coming from approximations in a model) and volitional uncertainty (arising from decision making). In addition, the work of Thunnissen (2005) demonstrated also that various definitions of uncertainty co-exist in engineering itself. Some disciplines such as control systems view it essentially as errors between models and reality while

others, such as the computational analysis and simulation community, view uncertainty simultaneously as variability, lack of knowledge and incomplete information in addition to errors (Thunnissen, 2003). The topic is therefore far from consensual across different sciences and applications.

Various authors have also defined and classified uncertainty within engineering design. McManus and Hastings (2004) viewed uncertainty as lack of knowledge, lack of definition, random phenomena, known unknowns and unknown unknowns. *Lack of knowledge* was related to facts needed to complete a design but “that are not known or are known only imprecisely”, *lack of definition* was seen as “things that have not been decided or specified” and *random phenomena* was defined similarly to aleatory uncertainty (McManus and Hastings, 2004). They defined also *known unknowns* as uncertainty designers are aware of and that may be characterized with time and/or effort using statistical analysis, while *unknown unknowns* is related to future events which can be predicted in advance (e.g. an asteroid striking a space system).

Another view was also provided by Earl et al. (2005), who classified uncertainty in design into four types: known uncertainties, unknown uncertainties, uncertainties in the data and uncertainties in the description. Uncertainty of data included lack of completeness, accuracy or consistency, including in measurements, that affect the design activity, while uncertainty in the description referred to ambiguity and lack of clarity in system definition (Earl et al., 2005). Known and unknown uncertainty were viewed similarly to the known and unknown unknowns of McManus and Hastings (2004). Moreover, a subsequent work by De Weck et al. (2007) classified uncertainty relevant to system design according to its source into *endogenous* and *exogenous* uncertainty. According to this taxonomy, endogenous uncertainty relates to factors within the domain of the system itself, such as uncertainties concerning product technologies and reliability or the corporate’s strategy and its business relationships. On the other hand, exogenous uncertainty relates to those which are outside of the companies’ control, such as uncertainties concerning the needs of users, political or regulatory stakeholders (De Weck et al., 2007).

The existence of a wide range of definitions and taxonomies across sciences and within

engineering design suggests that the meaning of uncertainty is elusive and consequently it is interpreted differently according to the particular context under consideration.

#### **4.2.1 Scope and an uncertainty definition**

Based on the previous observations and similarly to Zimmermann (2000), this dissertation views uncertainty as a *situational* property that depends upon the human's subjective interpretation of the quantity and quality of the information available describing a particular phenomenon or system of interest. Being a situational property, the perception of the “types” and “causes” of uncertainty thus varies with the surrounding context. As suggested above, this interpretation of uncertainty explains why a wide range of taxonomies has been found across the different sciences. Moreover, this view suggests also that a universal definition is not possible and that uncertainty quantification methods need to be adjusted to the situation. The author thus agrees that the choice of the quantification technique should take into account the type, quantity and quality of the information available and the language required by the user who will interpret the meaning of uncertainty (Zimmermann, 2000).

It thus follows that when referring to an uncertainty definition and taxonomy the context of analysis needs to be bounded and well-defined. This chapter addresses the issue of uncertainty management in complex product development. Furthermore, the author's research specifically approaches situations in complex design processes where requirements or other design variables of uncertain value – in the sense that they are expected to change – are communicated as inputs to activities performed by multiple interdependent design teams. The author is thus only concerned with the “types” and “causes” of uncertainty associated with the prescription of values to these design variables.

Considering the previous scope, this thesis defines uncertainty as absence of sufficient knowledge, definition or confidence in design variables that prevent design process participants to specify or predict a priori the system being designed – its behavior and properties – in a deterministic and quantitative way. There are two key ideas in this definition. The first follows from the work of McManus and Hastings (2004) and is that uncertainty arises

from lack of knowledge, definition and confidence about the value assigned to design variables. The second is that this uncertainty inhibits a deterministic approach to the design process, even in cases where the new design is largely based in incremental modifications made to a past and successful product. The following sections provide the author's perspective about the three main types of uncertainty affecting the prescription of values to design variables – variability, model uncertainty and imprecision – based on prior work reported in design literature.

### 4.2.2 Variability, aleatory and stochastic uncertainty

Variability, aleatory uncertainty or stochastic uncertainty are alternative terms which have been used to designate the amount by which a design variable is expected to vary about its *nominal* value as a result of processes that engineers do not fully control or comprehend. It relates to lack of knowledge about uncontrolled factors, such as manufacturing processes, material properties or operating conditions. A common example is changes in part dimensions as a result of variability in manufacturing processes. Other examples include, for instance, changes in expected component life due to variability in raw material properties or in-service loads.

Since variability affects the performance of technical systems, a wide range of methods have been developed to make systems less sensitive to such sources of uncertainty. Probability-based methods were among the first techniques to be used to accommodate noise in the design of systems for increased reliability and safety (Cornell, 1969; Kapur and Lamberson, 1977). Robust design methods (Tsui, 1992) typically address the problem of finding values for design variables that satisfy requirements despite the existence of uncontrolled noise factors (Chen et al., 1996). And multi-objective optimization methods have been extended to incorporate variability into objective functions and constraints that guide the solution search algorithms (Sundaresan et al., 1995; Das, 2000; Messac and Ismail-Yahaya, 2002). These modifications led to robust design optimization methods that have been applied to relevant industrial problems (e.g. McAllister and Simpson (2003); Poloni et al. (2006)).

### 4.2.3 Model uncertainty

Model uncertainty refers to the amount by which the true value of a design variable differs from the value *computed* for it. Analysis and decision-making in engineering design is normally supported by some type of mathematical model (Goh et al., 2007). These models contain uncertainty, since all embed mathematical and physical simplifications of reality and numerical and programming approximations.

There are many examples in the design context. Mathematical and numerical simplifications in gas turbine engine design are required, for instance, due to incomplete knowledge about complex physical processes such as combustion and turbulence (Saravanamuttoo et al., 2001; Mattingly et al., 2003). All researchers and practitioners involved in modeling and simulation have traditionally addressed the issues of model calibration, validity and accuracy (Arendt et al., 2012) and thus the issue is transversal to all disciplines used in integrated product development environments, such as structural, aerodynamic, thermal and control systems engineering.

Another issue that arises is to understand and match different levels of model uncertainty arising from multi-fidelity predictions (Sinha et al., 2013). The design process normally evolves from early estimates of design variables based on rough models to more accurate computations of their values based on complex models during later stages. However, situations occur in design processes when higher fidelity models are used earlier in conjunction with lower fidelity models. Since the former receives data inputs which are expected to contain higher model uncertainty, tuning the end result's fidelity is a considerable challenge even for model experts. In addition, experimental models used in physical validation of design features and behaviours and during prototype development also contain uncertainties of the same type.

### 4.2.4 Design imprecision

The foundational work of Wood et al. (1990), Otto and Antonsson (1994) and Antonsson and Otto (1995) identified and defined another form of uncertainty in engineering design: design imprecision. This term expresses variations in the values assigned to de-



sign variables due to updated *preferences* of the designer. The design process involves choosing values for such design variables within permissible ranges of the design space. To a reasonable extent, this process is similar to a decision-making process of choosing among alternatives. Imprecision thus expresses uncertainty in decision-making. It is the uncertainty surrounding the preferred value for a design variable, which is seen to *emerge* from the design process (Otto and Antonsson, 1994).

This form of uncertainty is known to be highest at the preliminary design stage (Wood et al., 1990). During this stage, requirements have not solidified yet and often change; engineers have more freedom to switch between alternative values for design variables; and design teams explore concurrently several concepts for the system, sub-systems and components. Real options analysis has been a technique recently explored to introduce flexibility in a system's architecture or in some components to mitigate the effect of such uncertainties during the development of some complex technical systems, such as large infrastructural projects (de Neufville and Scholtes, 2011).

Design iterations – which are normally faster during this stage due to the use of lower fidelity design tools – gradually reduce imprecision in design variables during the process and allow participants to gain knowledge and confidence about what is the best solution to choose among the alternatives. This imprecision reduction process is illustrated in Figure 4.1. By the end of the preliminary design stage, engineers have typically committed to one concept (down selected among alternatives) which is delivered to the detailed design stage. Model uncertainty and variability are also represented in Figure 4.1, showing how the true value may differ from the value specified to the variable during later stages of product development.

The work of Wood et al. (1990), Otto and Antonsson (1994) and Antonsson and Otto (1995) pointed also original research directions through the use of fuzzy set theory (Zadeh, 1965) to quantify uncertainty arising from imprecision. Their contribution pioneered the use of new mathematical concepts meanwhile introduced in evidence theory (Dempster, 1968; Shaffer, 1976), interval analysis (Moore, 1966), possibility theory (Dubois and Prade, 1988) and imprecise probability theory (Walley, 1991) in engineering design research. Sub-

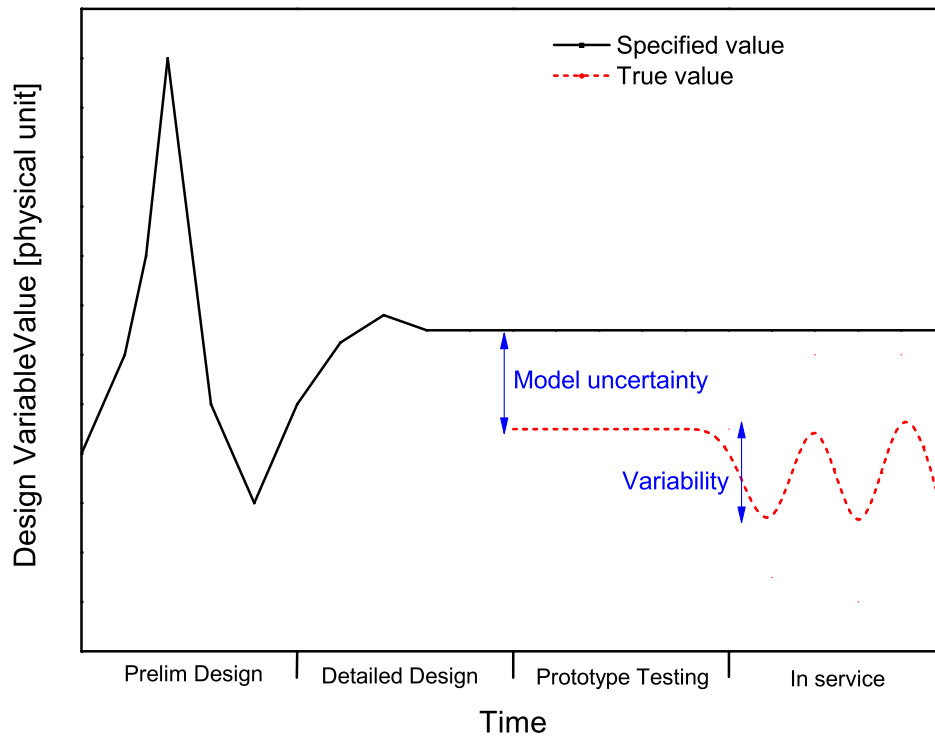


Figure 4.1: Uncertainties surrounding the value of a design variable during product development.

sequent literature explored such concepts. Cao and Rao (2002) investigated the use of interval analysis for the performance prediction and design optimization of a brake system. Designing for safety and reliability were examined by other authors through the use of possibility theory (Nikolaidis et al., 2004; Mourelatos and Zhou, 2005). Aughenbaugh and Paredis (2006) argued that when uncertainty is large the use of imprecise probability theory generates designs with higher expected utility than the traditional probability approach.

#### 4.2.5 Research opportunities

Experience with industry shows however that these mathematical concepts remain relatively unfamiliar to the engineers involved in practice in design activities. This is probably why, to the author's knowledge, prior work has remained essentially confined to academic design examples (Cao and Rao, 2002; Nikolaidis et al., 2004; Aughenbaugh and Paredis, 2006). This thesis claims that this constitutes a research gap and thus there is room for alternative approaches to design imprecision quantification and communication that show

its applicability in industrial environments.

In addition, the author argues also there are other gaps arising from the traditional expression of imprecision as variations in the preferences of individual designers (Antonsson and Otto, 1995). In fact, the development of large technical systems requires individual agents to engage into collaborative and distributed design processes (Movahed-Khah et al., 2010). Distributed design refers to the division of responsibilities and tasks across a range of design teams, where each is concerned with delivering a part of the whole solution – a sub-system, a component or a disciplinary aspect of the design – and uses its own tools and knowledge to achieve its goals (Ostrosi et al., 2012). Collaborative design refers to the interactions between design teams, which comprise information and knowledge exchange to drive design activities and negotiation to reach agreements (Shai and Reich, 2004) about requirements, design solutions, modeling approaches, evaluation criteria or assumptions about data, for instance. Due to the socio-technical interactive nature of collaborative and distributed design, understanding and quantifying the fluctuations in the preferences of *design teams*, namely in design variables which are communicated regularly between a team to be used as inputs to activities by another team or variables shared at interfaces requiring common agreement, is also of great importance.

Several issues arise from the *extension* of imprecision to variations in the preferences of a group of designers forming a team. The first question is the process followed by designers within a team or across teams to depart from their own individual preferences – which may conflict – and arrive to common or shared preferences about the value to assign to design variables. This relates to design consensus and design conflict resolution in collaborative and distributed design. Both topics have been object of various research approaches by Adelson (1999); Lu et al. (2000); Xiao et al. (2005); Ouertani (2008), for instance. Another issue is the quantification and modeling of the uncertainty surrounding shared preferences. Shared preferences may depend on the degree of consensus achieved by individuals (Barron et al., 1998) or be influenced by a strong authority – a supra decision-maker – existing in the team (Hazelrigg, 1996). In addition, the work of Wardekker et al. (2008) shows that it is critical to find systematic approaches to com-

municate uncertainty in ways that are easy to understand and minimize differences in interpretation of its meaning by individuals in a group. The same issue applies to the communication of imprecision arising from changes in preferences of design teams.

Considering these opportunities for research, this work intends to complement and extend prior literature on uncertainty in engineering design with the proposal of a novel method for imprecision management targeting industrial practice. The focus on a practical approach supporting designers and engineers aims to fill one of the gaps identified in prior literature. Moreover, the method proposed in this thesis constitutes also a mean to understand and quantify variations in the preferences of design teams – and not only of individual designers – and provides a visual form of communicating imprecision to design teams with the intent of promoting a shared interpretation of its meaning by designers.

## **4.3 A method for imprecision management**

This section presents the five main steps included in the proposed method for understanding, quantifying and communicating uncertainty levels under the form of imprecision during product development projects. The author coined the method as the **Imprecision Management Method (IMM)**. Figure 4.2 illustrates the method and supports the discussion throughout the course of the following subsections.

### **4.3.1 Collection of historical records**

The IMM proposed in this dissertation starts with the collection of historical records characterizing the behavior of uncertain design variables which are of interest to the organization from past product development projects. The author proposes to characterize the level of imprecision in design variables using past records of change found in these variables. It is argued that design variables change during the design process to reflect the evolution in design choices and preferences of individual participants or design teams. Figure 4.2 represents this initial historical data collection stage as step 1 in the IMM. It also shows that historical information about design variables is normally recorded in

various project documents produced during product development, such as requirements documents, communication records, engineering memos and design reports (Figure 4.2). Each organization thus contains an historical body of knowledge about the design variables and past changes in such project documents. The goal of the IMM's first step is to capture it.

### 4.3.2 Reconstruction of the variables' time evolution

The following step of the IMM uses the historical records retrieved from past project documents during the previous stage to reconstruct the time evolution of the uncertain design variables that were selected by the organization. This stage is represented in Figure 4.2 as step 2 and allows practitioners to identify, understand and group the changes that occurred in each of the variables under study. The analysis of the design variables' time evolution may additionally allow the division of the project's duration into different time periods, according to how the level of uncertainty in the variables is seen to evolve over time. The output from step 2 of the IMM is the full set of changes identified for each variable and grouped into a sample of changes that is eligible for statistical analysis, as shown in Figure 4.2. Since the goal of this stage is to produce unbiased and statistically significant samples of change for each design variable, it is important that historical records are collected from *multiple* development projects completed by the organization.

### 4.3.3 Statistical characterization of imprecision

The third step in the IMM is to perform a statistical analysis to the sample of changes found from several multiple past projects for each design variable under study. The author views the level of change that occurred during a project as a *proxy* of the underlying level of design imprecision. As a result, this thesis proposes that the level of imprecision in a design variable is inferred from two key change indicators:

1. The **Magnitude of Change (MoC)**, i.e., the amount which represents the shift in preference at a particular project time relatively to a value previously assigned to the variable. Figure 4.2 illustrates that the statistical characterization of the MoC

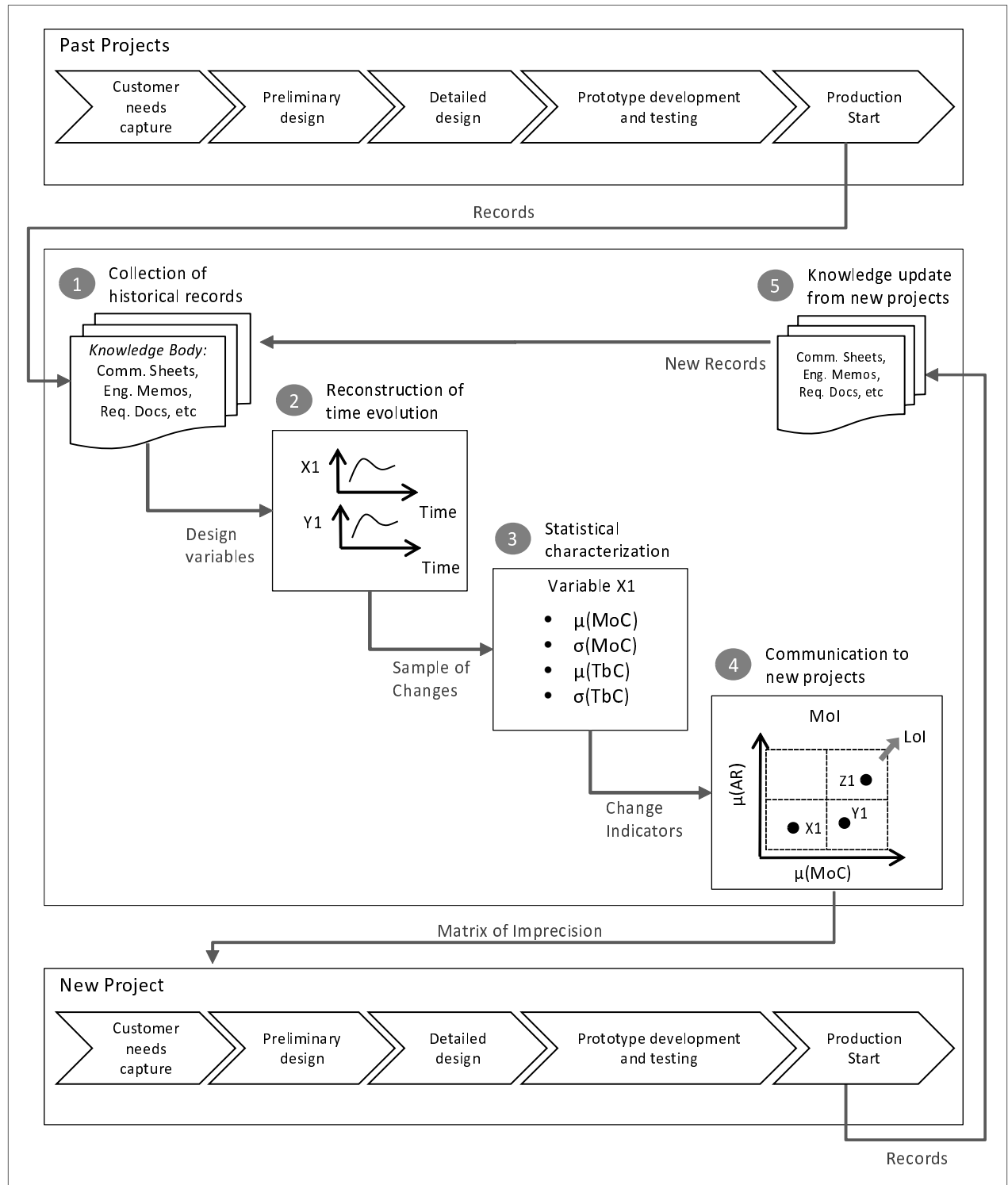


Figure 4.2: The Imprecision Management Method. Legend: Comm. - communication; Eng. - engineering; Req. Docs - requirement documents;  $\mu$  - arithmetic average;  $\sigma$  - standard deviation; MoC - magnitude of change; TbC - time between changes; AR - arrival rate of change; MoI - matrix of imprecision; LoI - level of imprecision.

for a generic variable  $X1$  can be performed using, for instance, the average amount of change ( $\mu(MoC)$ ) and the standard deviation ( $\sigma(MoC)$ ).

2. The **Time between Changes (TbC)**, i.e., the span observed between two consecutive decisions of changing the value assigned to the variable during a project. The average and standard deviation are proposed in Figure 4.2 to characterize statistically the time between changes. In addition, the inverse of TbC – which represents an *arrival rate of change* – can also be used as an alternative indicator.

These change indicators are proposed as proxy variables for imprecision and the IMM performs its characterization using traditional statistical measures which are familiar to designers and engineers involved in design practice, such as average values, standard deviations, probability density functions, etc. Although Figure 4.2 describes stage 3 of the method using a statistical characterization based on the arithmetic average and standard deviation, other statistical measures such as the median or the coefficient of variation can also be considered by the industrial users who will interpret and communicate the meaning of this form of uncertainty. The proxy variables proposed in the method – the MoC and the TbC – are treated as random variables, which is an underlying assumption in the approach.

#### 4.3.4 Communication of imprecision to new projects

The fourth step in the IMM consists in the communication to new projects of the *typical* level of imprecision surrounding design variables that should be expected by designers and engineers involved in new product development projects. Change indicators obtained from the previous step of the method are used as inputs to step 4 in Figure 4.2 with the purpose of communicating the meaning of design uncertainty under the form of imprecision in a language familiar to engineers. The indicators proposed in the method are used to bound the level of imprecision that can be expected. For instance, the typical level of imprecision in design variable  $X1$  to be expected can be characterized as the interval  $X1 \pm \mu(MoC(X1))$ .

Moreover, to promote a visual representation of the typical level of imprecision surrounding design variables – which facilitates communication in large organizations realizing complex systems – the author proposes the use of a **Matrix of Imprecision (MoI)** which joins the average amount of change and the average arrival rate of changes that should be expected during product development (Figure 4.2). The MoI is read similarly to a risk matrix: variables with higher arrival rates and higher magnitudes of change denote an overall higher Level of Imprecision (LoI). The level of imprecision characterizing a set of critical variables for the organization – for instance, the generic set  $X1$ ,  $Y1$  and  $Z1$  illustrated in step 4 of Figure 4.2 – can thus inform designers and engineers engaged in a new project about what is the typical level of uncertainty that is likely to be encountered.

#### **4.3.5 Knowledge update from new projects**

The last step proposed in the method is a continuous update and refinement of the historical body of knowledge about design variables that supports it. Step 5 in Figure 4.2 illustrates that completion of a new product development project triggers an action of knowledge update with new information about the behaviour of the design variables during that project. This ensures a continuous increase in historical records and, most importantly, an increase of the size of the samples of changes. Therefore, steps 1 to 4 described in Figure 4.2 shall be repeated in subsequent analysis with a larger data set which leads to increasing accuracy in predictions with the progressive use of the method. This is an important step in the method, since the IMM relies essentially on inference: future levels of imprecision are inferred from past observations contained in samples of changes that have been collected.

### **4.4 Empirical findings from an industrial case**

This chapter also reports empirical results from an industrial case-study performed by the author at Rolls-Royce where the method proposed was tested. As mentioned in Section 4.1, the purpose was to exemplify and test the method in practice and demonstrate its



potential. This section presents and explores the findings that were generated during the research. It begins by describing the context and the design variables selected for the case-study and then presents the findings.

#### 4.4.1 Setting

As explained in Section 3.2, the author was allowed to perform research at Rolls-Royce about the design processes followed to design new gas turbine engines for aerospace applications. This section introduces and explains a simplified and high-level representation of the design process used to realize a new engine. This representation is contained in Figure 4.3, based on literature (Saravanamuttoo et al., 2001; Mattingly et al., 2003) and the author’s empirical research. Since the author was interested in using the IMM to understand, quantify and communicate the typical levels of imprecision to be expected during new projects, the research focused in the *preliminary* design stage, since it is known that design imprecision is higher during that period (Wood et al., 1990).

It was found that the engine’s preliminary design process is particularly dynamic. Figure 4.3 illustrates that project teams at Rolls-Royce depart either from a Request-for-Information (RFI) or a Request-for-Proposal (RFP) from a customer interested in a new system envisioned for a particular market application. Customer requirements are captured from these requests and trigger the start of system design activities to respond to the customer’s request (design loop 1 in Figure 4.3). Furthermore, the company’s market research team is also active and surveys the needs of the final users – the airlines – and the competitors’ strategies.

The set of requirements for design and development are captured from these interactions both with the customer and the market. These contain high level requirements such as thrust, specific fuel consumption, weight, unit cost or reliability (Figure 4.3). A team responsible for the design of the whole system then performs the design of the Thermodynamic Cycle at different operational points (take-off, climb, cruise, etc) that form the aircraft’s mission. The engine must be capable of generating the required thrust within the allowable fuel consumption, weight and cost targets. The turbomachinery’s prelimi-

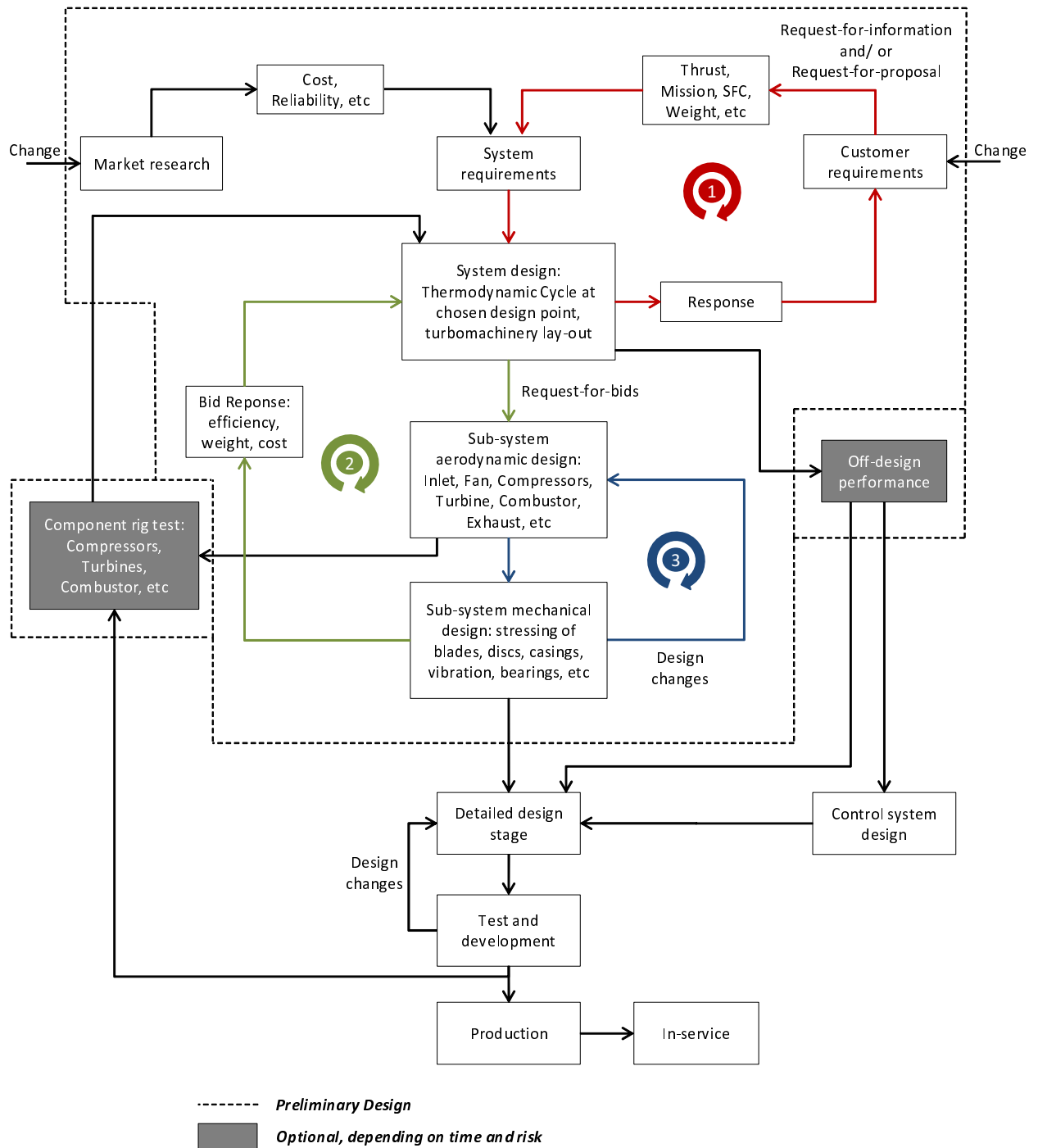


Figure 4.3: High-level view of a gas turbine design process, adapted from Saravanamuttoo et al. (2001) and Mattingly et al. (2003) based on the author's research at Rolls-Royce. The activities normally performed during the preliminary design are enclosed by the dotted line. Legend: 1 - design loop involving the customer and the system design team; 2 - design loop involving the system and sub-system design teams; 3 - design loops within each sub-system involving aerodynamic and mechanical component design and integration.

nary mechanical arrangement is also selected at this point. When this team is satisfied with a design solution, the cycle and the turbomachinery lay-out is sent as functional requirements to all the sub-system teams responsible for designing the Fan, the Compressors, the Combustor, the Turbines and the remaining sub-systems. Sub-system functional requirements are sent under the form of a request-for-a-bid which triggers the design loop 2 represented in Figure 4.3 involving both system and sub-system design teams.

Each of the sub-system teams then engages in inner design loops encompassing the aerodynamic and mechanical design of the sub-system and its key components, such as blades, discs and casings. Figure 4.3 illustrates it as design loop 3. When component designs are integrated successfully into a sub-system solution that fulfills the functional requirements received, teams bid the outcome of their design activities back to system design. Such bid will contain the design solution obtained and relevant parameters - such as component efficiencies - that have been estimated by the team designing the engine's cycle to meet customer requirements in design loop 1 identified in Figure 4.3.

This process then restarts until a solution is reached that satisfies all the participants involved and the needs of the customer and final users that are expressed in the system requirements. Some activities such as off-design performance or component rig testing may be performed or postponed for subsequent development stages, depending upon the time available to reach a design solution and the level of risk involved. The values assigned to design variables change along the way to reflect updates in the *preferences* of the designers and engineers involved. Moreover, participants meet regularly to collaborate, solve design challenges and to agree upon what is the best solution that can be technically achieved within the time available and the level of risk that can be tolerated. The final Proposal sent to the customer reflects the outcome of this collaborative process. Customer and market needs are uncertain and may also change during the process, as shown in Figure 4.3, and effectively trigger changes in high-level requirements as it has been found in the findings discussed during Chapter 3.

Based on this study of the preliminary design process, it was concluded that characterizing the typical level of uncertainty in variables exchanged between the system and

sub-system design teams would be of interest both to the manufacturer and to academia. Therefore, the author selected a set of 36 design variables that are normally communicated between these teams. Design variables have been divided into independent decision variables, intermediate variables arising from the processing of decision variables and performance variables, which are dependent variables used to measure the effectiveness of the design solution (Kusiak and Wang, 1995). The set selected in this study contains essentially independent decision variables under the control of system designers and intermediate variables arising from them. It comprises aerodynamic and thermodynamic decision variables – inlet mass flow, compressor pressure ratio or combustor temperature, for instance – as well as mechanical – the spools’ rotational speed, for instance – and geometrical independent variables – for instance, the fan bypass ratio. Together with intermediate design variables that arise from the setting of such variables, the whole set defines the system’s design solution to fulfill the customer’s performance requirements of thrust, specific fuel consumption, weight, cost, etc. The whole set of design variables is *flown down* and communicated to sub-system design teams as functional requirements that guide their aerodynamic and mechanical design processes, as illustrated in Figure 4.3.

#### **4.4.2 Collecting historical records**

Following the first step in the IMM, the investigation begun with archival research to build a database of change records about the 36 design variables selected: the system’s solution definition parameters flown down as functional requirements to sub-system teams, which use them as inputs in their aerodynamic and mechanical design activities (Figure 4.3). The author was allowed to consult various historical archives at Rolls-Royce, namely the databases of *Communication Sheets Reports and Records* which document all the design information officially exchanged between teams during projects. These are extensive databases, which normally contain thousands of documents for each project. The research approach selected one particular development program consisting of a new gas turbine engine family for aerospace applications. Two major development projects

– Project "A" and Project "B" – had been performed leading to new products in this family. Both projects were selected due to higher availability and accessibility to data and to the designers and engineers that had been involved in the projects.

The communication records were the primary source of information and evidences about design changes, but the author also researched in various other related project documents, such as Requirements Documents exchanged between teams which were referenced in the communication records. Such investigation intended to triangulate the evidences found in the database of communication records and increase the confidence in the data found, which is a typical concern in case-study research methods (Yin, 1984). Personal semi-structured interviews (Robson, 2002) to system and sub-system engineers involved in the projects complemented the data validation process. These interviews were performed to confirm the record's reliability - ensuring that observed changes communicated in records had been in fact considered in design activities - and to capture the design choices which triggered them along the way.

### **4.4.3 Time evolution of design variables**

The author's research subsequently extracted from the Communication Sheets Reports and Records and Requirements Documents all the changes enabling the reconstruction of the time evolution of each of the 36 design variables during these projects. This dissertation shall refer to these variables as X1, X2, X3, ... , X35, X36. More than 30 changes were found in each variable (the total amount across Projects "A" and "B").

Figure 4.4 presents the time evolution of two of such variables – X6 and X19 – during Project "A". X6 and X19 refer to a temperature and a mechanical design variable, respectively. Nineteen different values were assigned to these design variables during one of the projects. It illustrates also that most design changes occurred during the first 24 months of the project, which corresponded roughly to the duration of the preliminary design stage. It was observed in this particular project that several architectures and concepts were explored by Rolls-Royce engineers during this period.

Therefore, the behavior observed in Figure 4.4 can be said to be characteristic of the

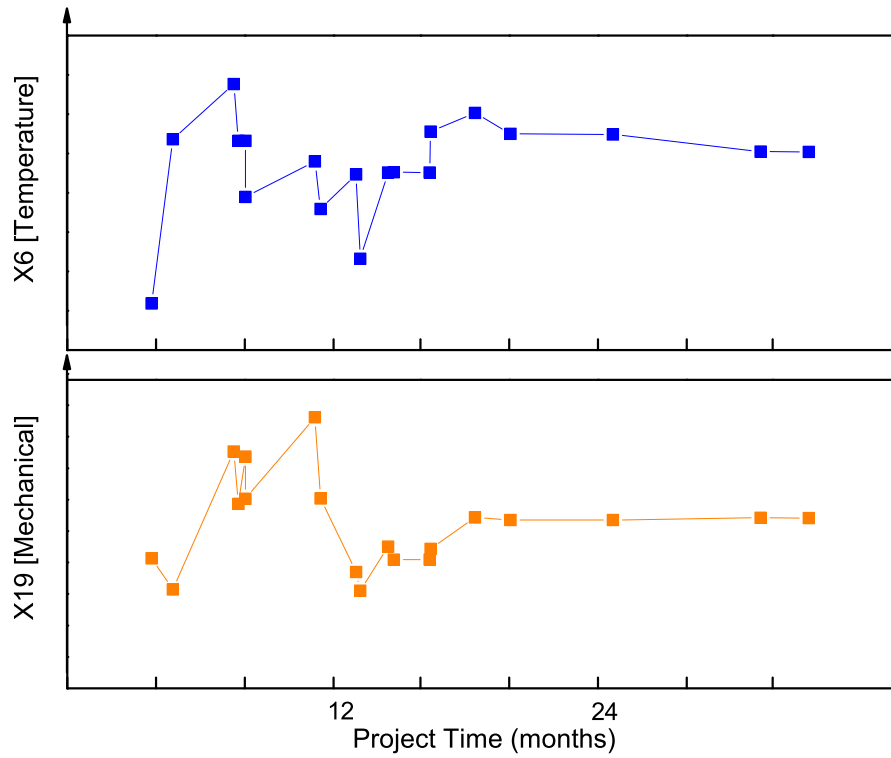


Figure 4.4: Time evolution of two design variables during Project "A" and for a particular engine operational condition. X6 and Y19 refer to a temperature and a mechanical design variable, respectively. Project time refers to the time since the project's start.

preliminary design. It also clearly supports that design imprecision is *higher* during this phase and that the preference of participants evolves and changes over time. This is related to the phenomenon of co-evolution of both requirements and solution definition which has been described in design literature by Maher et al. (1996) and Dorst and Cross (2001), for instance. The analysis of the variables' time evolution revealed that the first 12 months of the project were particularly dynamic and supported the division of the design process into different time periods. After the first 24 months, the value assigned to the design variables stabilized and thus it can be argued that design imprecision virtually disappeared after that time (Figure 4.4).

#### 4.4.4 Results from statistical analysis

This subsection brings forward the results from the application of the method's third step: statistical characterization of two proxy variables for imprecision which have been defined in the IMM as the magnitude of change and the time between changes (Figure 4.2). All

design variables were analyzed for a particular engine operational condition.

#### 4.4.4.1 Magnitude of change

Statistical characterization was performed based on *consecutive* observations. The magnitude of change, for instance, was computed from the time evolution of the design variable based on:

$$MoC(X1)_t = |(X1_t - X1_{t-1})/X1_{t-1}| \times 100 \quad (4.1)$$

where  $MoC(X1)_t$  refers to the magnitude of change of variable X1 at time  $t$ ;  $X1_t$  refers to the value assigned to design variable X1 at time  $t$  and  $X1_{t-1}$  refers to the value assigned to X1 at time  $t - 1$ .

Figures 4.5 and 4.6 show the results obtained from the application of Equation 4.1 to *different* periods of the design process. The time periods were chosen empirically, according to the time evolution found in some of the design variables studied during the previous step of the IMM. We have divided the preliminary design process into three periods: a) 0 to 6 months; b) 6 to 12 months; and c) 12 to 24 months. The approach allowed us to observe and characterize the level of imprecision during time. Such knowledge intends to support designers and engineers understanding and managing uncertainty during different stages of their design process, even during different periods of the preliminary design itself.

The results reported in Figure 4.5 demonstrate that the large majority of the design variables had an average magnitude of change lower than 6% in Projects "A" and "B". Furthermore, this Figure also shows that most retained an average MoC lower than 6% during the whole preliminary design. Figure 4.6 depicts the time evolution of the 36 design variables with more detail. It shows that most variables had an average MoC:

- Lower or equal to 4% during the first 6 months of the projects;
- Lower or equal to 3% during the second 6 months period (6 to 12 months);
- Lower or equal to 1% during the subsequent 12 months of the projects (12 to 24 months).

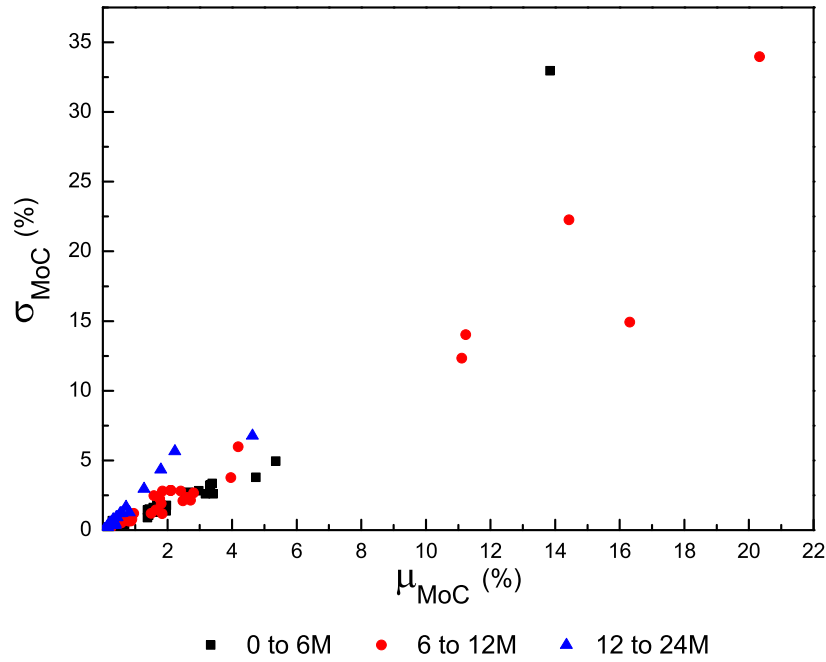


Figure 4.5: Average and standard deviation of the magnitude of change (MoC) found for the 36 design variables studied. Results are shown for three different time periods in the design process: 0 to 6 months; 6 to 12 months; and 12 to 24 months.

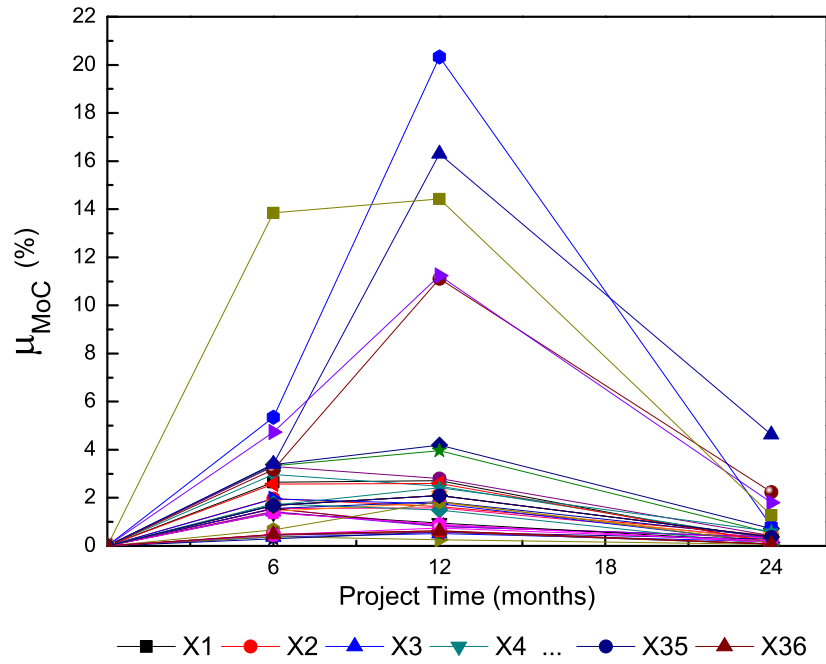


Figure 4.6: Time evolution of the average magnitude of change (MoC) found for the 36 design variables studied.



However, both Figures 4.5 and 4.6 demonstrate that some of the design variables had *very different* behaviours. Some were found to have a high average MoC – between 10% and 21% – particularly during the second 6 months period, i.e., 6 to 12 months after the start of the project (Figure 4.6). Such variables are also characterized by high standard deviations as shown in Figure 4.5. It is a clear indication that the MoC of change varied extensively during the design process and thus that this particular subset of variables is highly *uncertain*.

#### 4.4.4.2 Time between changes

The results obtained for the second proxy variable are presented in Figures 4.7 and 4.8. These Figures characterize the Time between Changes in a statistical sense for *all* of the 36 design variables studied in this particular case. These variables were communicated from the system's design team to the sub-system teams at the same time during the projects, in a batch form. Although that was not the case in these projects, *asynchronous* communication of design information between teams can be assumed as the most general case to be found in practice during complex product development. Asynchronous interactions have been in fact observed by the author in other situations during the field research performed at Rolls-Royce, namely during the communication of interdependent design variables among discipline engineers involved in sub-system and component design cycles.

The histogram found in Figure 4.7 for the TbC shows that design variables changed most frequently every 4 or 8 weeks during the projects investigated. Moreover, the cumulative probability distribution derived from this empirical histogram reveals that the probability of a TbC lower or equal than 8 weeks was approximately 67% in Projects "A" and "B" (i.e.  $P(TbC \leq 8) = 0.67$ ). In addition, the time evolution of the TbC is presented in Figure 4.8, from which it has been found that the average TbC was:

- Equal to  $\approx 4.6$  weeks during the first 6 months of the projects;
- Equal to  $\approx 5.7$  weeks during the second 6 months period (6 to 12 months);

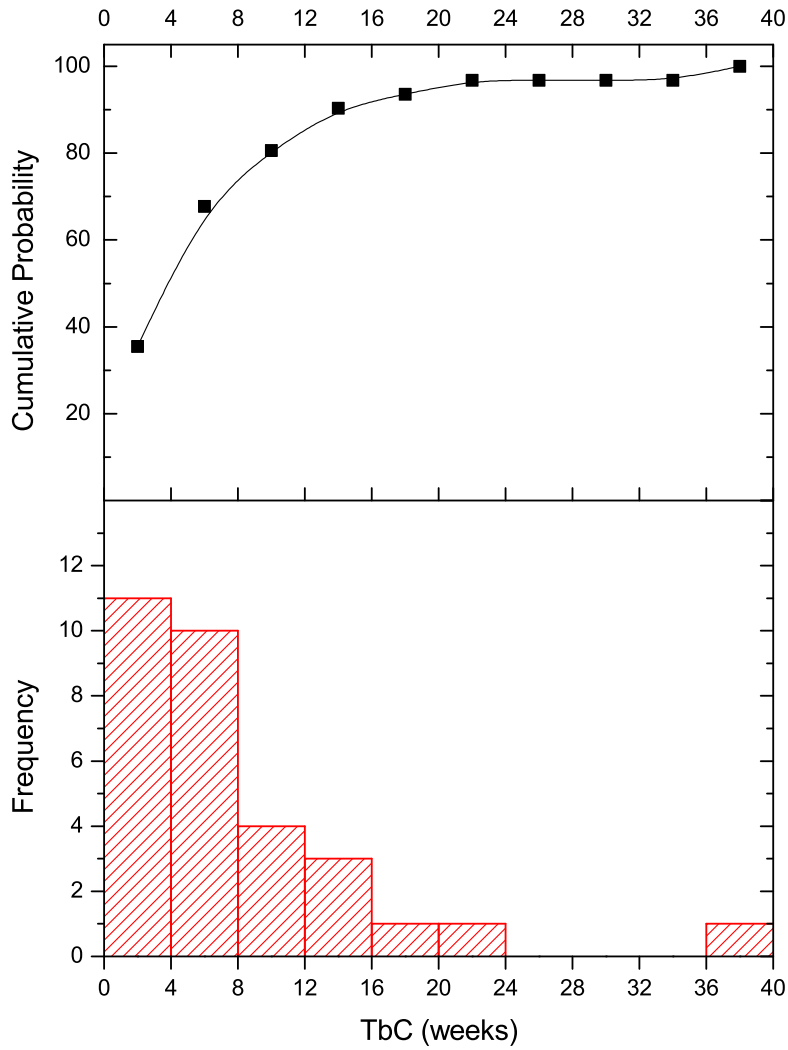


Figure 4.7: Histogram and cumulative probability distribution for the time between changes (TbC) found for the design variables studied.

- Equal to  $\approx 14.1$  weeks during the period comprised between 12 and 24 months after the projects' start;
- Equal to  $\approx 7.2$  weeks considering all of the preliminary design stage (0 to 24 months).

These are important findings for designers, engineers and managers at Rolls-Royce. They quantify the *period* of their design processes, namely in the iteration loop represented in Figure 4.3 that involves the system and sub-system design teams. Figure 4.8 demonstrates how the average TbC varies during the preliminary design. It can thus support design teams at Rolls-Royce organizing and planning design activities taking into account how the expected arrival rate of change evolves during complex product development.

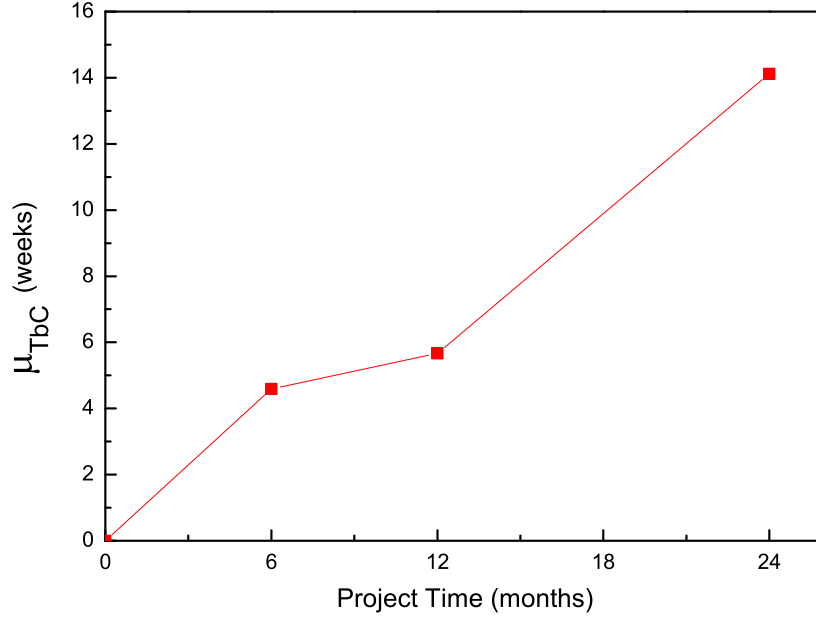


Figure 4.8: Time evolution of the average time between changes (TbC) found for the design variables studied.

#### 4.4.5 Communicating imprecision levels

The case-study performed at Rolls-Royce enabled the author to produce an example of the Matrix of Imprecision (MoI) that has been proposed in Section 4.3.4 and illustrated in Figure 4.2 as the fourth step in the IMM. The MoI presented in Figure 4.9 positions some of the design variables studied – X19 and X22 – according to their average MoC and average Arrival Rate (AR) of changes. The average AR is simply  $\mu(AR) = 1/\mu(TbC)$  which in this case gives a value of approximately 0.56 changes/month for both X19 and X22.

Considering that, in this particular case, the remaining design variables that were studied have the same average AR, additional illustrative data was added to Figure 4.9 to support the following discussion of the MoI's communication potential.

The MoI can contain, for instance, distinct imprecision *windows* to position variables according to higher or lower typical levels of imprecision. This dissertation presents 4 windows in Figure 4.9: low AR/ low MoC; low AR/ high MoC; high AR/ low MoC; and high AR/ high MoC. Design imprecision is naturally higher as we move in the matrix to windows of both higher arrival rate and higher magnitude of change. The idea can be

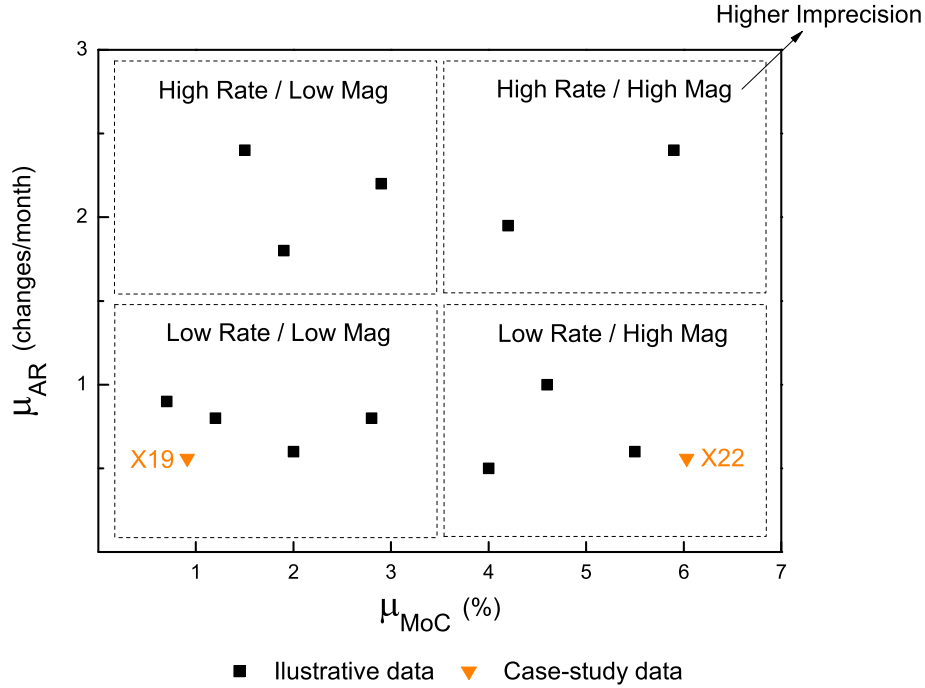


Figure 4.9: Matrix of imprecision showing the typical level of imprecision that can be expected in two of the design variables studied. X19 and X22 refer to a mechanical and a mass flow design variable, respectively.

naturally extended to other topologies that may be defined for this matrix.

According to the results found during the case-study and the simple low/ high topology that is defined in this thesis, Figure 4.9 locates:

- X19 in the window of low AR and low MoC;
- X22 in the window of low AR and high MoC.

The matrix thus shows in a visual manner that X22 is typically characterized by a level of imprecision higher than X19. The MoI allows engineers and designers to visualize and communicate a comparison between requirements and other critical design variables in terms of its typical levels of imprecision. Project participants at Rolls-Royce designing sub-systems and components that are affected by changes in the functional requirement X22 may be informed of the expected AR and MoC through the matrix presented in Figure 4.9. Using this information they may decide to delay the start of new design iterations when a new change is expectable (according to the expected TbC or AR). Or they may decide to perform sensitivity studies and design for higher robustness to change in

order to accommodate changes below a chosen *threshold*. The threshold of change may be defined as the average MoC, for instance. However, it is left essentially up to the designer to decide upon what is the adequate threshold, according to his expert knowledge of the sub-system or component design solution's sensitiveness.

Furthermore, the MoI can also present how the imprecision level of a particular design variable evolved during different stages of the product development process or even be used to compare imprecision levels between different projects performed across the same organization. Joining quantitative and qualitative information into a single representation gives the tool flexibility to tackle various types of information analysis. Considering that the MoI is also capable of acting as a repository of past data, this dissertation argues that it can be used effectively as a knowledge management tool in development intensive industrial organizations.

## 4.5 Discussion

The original results resulting from the method's application at the complex systems manufacturer supporting this thesis allowed also a critical analysis and discussion of the method's strengths and limitations, including its scope of application. This section presents such analysis and discussion. Furthermore, the author's view about how the proposed method can be integrated with other approaches during design process management is also presented.

### 4.5.1 Strengths

Case-study findings presented in Section 4.4 show that the Imprecision Management Method proposed in this Chapter has several strengths. The most important one is its relative simplicity when compared to prior work based on advanced mathematical concepts (Cao and Rao, 2002; Nikolaidis et al., 2004; Aughenbaugh and Paredis, 2006) which makes it easier to apply in industrial practice. The author's ability to apply the method

at a major manufacturer and report interesting results – despite the fact that academic researchers may have limited knowledge about the product and access to project archives – suggests that the method can be replicated systematically by company engineers with greater knowledge and access than researchers.

The second strength is the method’s ability to communicate uncertainty during early design arising from the statistical characterization of imprecision based on change indicators which are simple to compute and interpret by designers and engineers. Results from the case-study presented in section 4.4 demonstrate also that the IMM can quantify and communicate imprecision arising from the evolution of preferences of design teams, since project archives often contain communication records between teams that are sources of data. Understanding the evolution of shared preferences had been identified in Section 4.2.5 as an opportunity for research. However, it should be noted that the IMM does not tackle the *process* followed by a group of designers to arrive to a shared preference and resolve underlying design conflicts.

Moreover, considering the problem of mismatch in the interpretation and use of uncertainty information by the different individuals of a group reported by Wardekker et al. (2008), this thesis argues that the MoI promotes a shared understanding about the meaning of imprecision among designers and engineers. The author’s view is that this strength arises from the MoI through the combination of quantitative indicators / metrics – such as the Magnitude of Change and the Time between Changes – which are familiar to engineers and the use of qualitative topologies that facilitate a consistent distinction about the relative level of uncertainty among design variables.

### **4.5.2 Limitations**

The predictive power of future levels of uncertainty is the method’s main limitation, since it relies on the collection and analysis of past data about the behaviour of design variables. However, the knowledge update step included in the method – feed back data arising from new projects – ensures a continuous increase in historical records and in the size of the samples of change and thus increasing accuracy in predictions with the progressive use of

the method.

The accuracy of the method will vary according to the *amount* and *type* of data collected from past projects. A dataset comprising essentially a strong portfolio of projects characterized by incremental modifications shall be effective at predicting future levels of imprecision during designs with similar product architectures. However, its predictive capability can be reduced in a new project where major architectural changes are implemented. Conversely, historical records collected from research across a portfolio that comprises several product architectures and several past projects of each type will provide a richer database. In this case, the IMM can be used to compare how uncertainty varied within a set of projects sharing the same architecture and also across projects where there has been a major architectural change and support extrapolation to a new project with some level of confidence.

The scope of application of the method thus depends upon its continuous use and can be expected to increase as records collected from new projects are added to the main body of knowledge. An extreme case is that of radical changes in both the component technologies and the architecture linking them. In such cases, historical data about the behaviour of design variables is likely to be inadequate for prediction purposes. However, even such extrapolations may be used in practice due to lack of better information at a particular project time.

### **4.5.3 Integration with other change management approaches**

An additional opportunity that emerges from the findings presented in Section 4.4 is the integration of IMM with other approaches supporting change management. Several authors have proposed change propagation methods – relying on the capture of dependency information – to analyse the effects of component changes on the overall system’s architecture, such as the approaches introduced by Clarkson et al. (2004) or Giffin et al. (2009). These have been introduced and analysed in Chapter 2. The original proposals have also been subsequently extended to multidomain change propagation analysis and several authors have attempted to understand the propagation of changes in requirements

on the product's architecture (for instance, Hamraz et al. (2012) or Koh et al. (2012)). This thesis argues that the IMM may be used by designers and engineers as a source of data to perform multidomain change propagation analysis. Change propagation normally relies on the estimation of the probability and the impact of a change in one element affecting other elements through structural dependencies present in the domain(s) of analysis. Section 2.4.4 explained the governing principle in the context of requirements change impact assessment. The view defended in this dissertation is that the key change indicators produced by the IMM based on historical records can provide additional and reliable evidences to estimate or model both the probability and the impact of a change. For instance, the TbC retrieved from the IMM may be used to model the probability of changes in requirements or other design variables as a function of product development time, thus taking into account that the likelihood of change is higher in early development stages and then gradually reduces over time. And the impact of an *initiating change* (Clarkson et al., 2004) – from a requirement or other design variable – on subsequent changes across the domain(s) may be modeled also as a function of the known evolution of the MoC that arises from the IMM.

Therefore, the author's view is that the IMM and multidomain change propagation methods are *complementary* and may be integrated to support change management. Based on the limitations of change propagation methods discussed in Table 2.2, one particular benefit which may be anticipated from such integration is the reduction in the level of commitment required from experts to build and maintain models for change impact assessment. The author's belief is thus that such integration opens the possibility of further research on the topic.

## 4.6 Review of progresses

The current chapter tackled the second set of research questions identified in Section 1.4, which were primarily concerned with finding pragmatic approaches to support organizations predicting and managing the level uncertainty that should be expected during complex development projects. Various research progresses were made through the syn-



thesis of a new method and the realization of an industrial case-study to test the method's applicability in practice that allows the author to answer the research questions in the following manner:

**Q2** *How can the level of uncertainty in requirements be predicted and managed in industrial practice?*

The author recommends a structured approach to the prediction and management of imprecision – uncertainty in design arising from updated preferences – based on five steps: 1) collection of historical records about uncertain variables from past projects; 2) reconstruction of the design variables' time evolution using the records collected; 3) statistical characterization of design imprecision based on two key indicators, the Magnitude of Change and the Time between Changes; 4) communication of imprecision levels to new projects based on a Matrix of Imprecision, which relates the average magnitude and arrival rate of changes; and 5) continuous knowledge update from new projects to ensure increasing accuracy of the method. The approach proposed was purposely practical – relying on traditional statistical measures familiar to the recipients who will interpret the meaning of uncertainty – in order to facilitate its implementation in industrial organizations and to complement techniques reported in prior literature. It was coined as the Imprecision Management Method and aims to support engineers and designers understanding, quantifying and communicating uncertainty levels during complex product development.

**Q2-a** *What are the main types of uncertainty in engineering design?*

The review of literature showed that the general meaning of uncertainty is still a topic of debate and various taxonomies can be found across sciences. In the context of engineering design, the author views uncertainty as lack of sufficient knowledge, definition or confidence in design variables preventing engineers and designers to specify or predict a priori the system being designed in a deterministic and quantitative way. Furthermore, the author showed that variability or aleatory uncertainty, model uncertainty and design imprecision are the key types of uncertainty affecting the design activity. Variability or

aleatory uncertainty denotes the amount of variation expected as a result of random or uncontrolled processes. Model uncertainty is used to refer to variations in the values computed from engineering models due to mathematical, physical or numerical approximations. And imprecision was coined in prior literature to deal with variations arising from evolving preferences of engineers and designers during the design process.

**Q2-b** *How does the level of uncertainty in requirements and design variables varies during the course of complex product development projects?*

The imprecision management method proposed and applied in an empirical case-study revealed that the level of uncertainty evolves considerably during time. The case-study findings support that it is much higher during the preliminary design stage of complex product development. This stage took place during approximately 2 years in the projects investigated at Rolls-Royce. After this period, the level of imprecision virtually disappeared and this was an empirical result that is consistent with previous arguments and theoretical models proposed in design literature (in Wood et al. (1990), for instance). Considering the indicators proposed in the Imprecision Management Method to characterize the level of uncertainty, the research showed also that most variables studied in the case-study – functional requirements flown down to sub-system design teams – evolved from an average Magnitude of Change lower or equal to 4% during the first 6 months of the project to 1% approximately 24 months after the project's start. In addition, it was also shown that the average Time between Changes changed from approximately 4.6 weeks (observed during the first 6 months) to 14.1 weeks (found during the period comprised between 12 and 24 months after the project's start).

**Q2-c** *What is the typical level of uncertainty that should be expected by designers and engineers in new projects?*

The research performed by the author demonstrated that engineers and designers participating in new projects can expect a decreasing Magnitude of Change and an increasing Time between Changes during early stages of complex system development. The typical level of uncertainty that should be expected is characterized by an average Magnitude

of Change below 4% and an average Time between Changes around 7.2 weeks during the preliminary design stage. However, the author's investigation also demonstrated that engineers and designers should expect subsets of variables to be much more uncertain – since they were characterized by an average Magnitude of Change between 10% and 20% and high standard deviations – thus requiring a different level of management attention during the development process.

## **4.7 Summary**

The research motivation for this chapter originated from the needs of designers and engineers involved in complex design processes. Design imprecision – a type of uncertainty related to decision-making during design – triggers many changes in the values assigned to requirements and other critical variables communicated between design teams and used as inputs for process activities, particularly during the early stages of the process. Designers and engineers must decide how to organize design activities and it was found they often do so without sufficient knowledge about the typical level of imprecision that they should expect in the inputs necessary to perform their activities.

In order to support designers and engineers understanding, quantifying and communicating this type of uncertainty, this chapter has contributed with a method synthesized by the author from empirical observations and archival research performed during the time spent on-site at Rolls-Royce. The Imprecision Management Method proposed is based on a pragmatic approach relying on the collection of historical records of change, time evolution reconstruction, statistical characterization of the typical levels of imprecision that should be expected, communication of uncertainty levels to new projects and continuous knowledge update.

The second contribution presented in this chapter was an empirical case-study performed at Rolls-Royce – an organization where design process management is very challenging – to demonstrate the method's applicability in real industrial environments. The author applied the Imprecision Management Method through the collection of records of change of 36 design variables across two major aerospace projects. Several key conclusions were

unveiled from this investigation, namely that:

- The level of uncertainty in requirements or other variables varies considerably during time. It was found that imprecision was higher during the first 2 years of product development, which corresponded roughly to the preliminary design stage. During this time period, the average Magnitude of Change gradually decreased while the average Time between Changes raised also in a gradual way.
- Most of the variables studied can be expected to have an average Magnitude of Change below 4% and change with an average Time between Changes of 7 weeks during the preliminary design of new development projects. However, the findings also suggest that particular variables are highly uncertainty and large magnitudes of change should be expected by engineers and designers.

In summary, the method proposed and the findings explored during this chapter have provided original contributions answering research questions Q2, Q2-a, Q2-b and Q2-c. The central question regarding how can engineers in organizations predict and manage the level of uncertainty in critical input information for activities – such as requirements – has been answered through the prescription of the Imprecision Management Method. The method's strengths are its relative simplicity, the ability to communicate uncertainty through indicators familiar to practitioners and the promotion of a consistent understanding of its meaning within teams. The predictive power of the method is its main limitation, since it relies on historical data. However, the author argues that this limitation can be overcome through the continued use of the method.

The author's findings arising from the case-study demonstrate it is feasible to recover the historical body of change knowledge from project archives and use it to understand, characterize and communicate uncertainty in industrial organizations developing large technical systems. The results from the initial descriptive evaluation of the method reported in this chapter suggests that the Imprecision Management Method provides useful support for design process management and thus has industrial value. The author's belief is that an active management approach to uncertainty shall generate long-term competitive benefits in these organizations arising from design process improvements.

## Part III

# Planning for change

All models are wrong, but some  
are useful.

—GEORGE P. BOX



# Chapter 5

## Modelling complex and uncertain processes

Industrial needs identified from exploratory research encompassed also understanding and quantifying the effects of requirements change in complex product development projects so that engineers and managers can take them into account during planning activities. The author argued in Chapter 1 that model-based simulation is a cost-effective research methodology to investigate cause-effect relationships of this kind in product development. It was argued that the complexity of large development projects, the availability of few data or its lack of accessibility and the time required to follow these projects make the application of other research methodologies difficult to address such research goals. Despite of many limitations and the caution that is necessary in the interpretation of results arising from model-based simulations, this thesis claims that it provides ways to do virtual experimentation from which quantitative and qualitative insights can be gained about the behaviour of complex systems. These insights can then support decision-makers analyzing various scenarios and making more informed decisions.

Based on the choice of a model-based research approach, this Chapter is focused in reviewing and discussing prior approaches reported in product development literature to model and simulate complex and uncertain design processes <sup>3</sup>. It thus addresses one of

---

<sup>3</sup>This chapter builds upon initial research published in Fernandes et al. (2013), which explored activity-based models to model and simulate uncertainty in complex design processes.

the thesis's research questions (Q3-a) and is concerned also with bringing forward a comparative discussion about each of the main types of model-based approaches previously reported in literature.

Discussion proceeds in five main sections. Firstly, the main purposes that have been identified in literature for product development modelling are introduced. Secondly, this thesis presents some of the main challenges in developing models of product development, focusing particularly in those specific to creating models of complex and uncertain design processes, which are normally found during the development of large technical systems. The author draws from empirical observations made during the time spent on-site at Rolls-Royce – investigating projects and interviewing many engineers and designers – to discuss key characteristics of complex design which need to be captured and represented in models. Thirdly, a comprehensive review of the different types of models previously investigated in literature is presented, together with a critical discussion about the strengths and limitations of each approach. The author finalizes with the review of research progresses and a summary of the chapter's main findings.

## **5.1 Purposes of process modelling**

The central purpose of product development process modelling is to provide planning support to organizations. The importance and difficulty of planning complex systems development has been recognized by many authors in literature. For instance, Eckert and Clarkson (2010) have shown recently that complexity and uncertainty in such products forces industrial organizations to produce and manage a multitude of plans in parallel during projects: product plans focusing on cost, on the bill of materials and in procurement issues; process plans targeting the control of milestones, lead-times or project activities; resource plans; and quality plans. Experience thus shows there are multiple facets of planning, each focusing in attaining a particular objective. Acknowledging the different dimensions of the planning activity, this thesis follows Browning and Ramasesh (2007) who recently identified and organized the *purposes* of product development process modelling into four categories:



- **Visualization** purposes, supporting engineers and managers to understand their activities, interactions and commitments from visual representations containing which work elements should be executed and when and how they should be done. It is argued that visualizing processes helps teams to focus and align project goals and promotes organizational accountability and collaboration during group discussions.
- **Process planning** purposes, supporting project participants to make commitments, select activities to be done taking into account uncertainties and risks and to structure the timing and sequence of activities and their successful integration. Process models with simulation capabilities support also the estimation of key project metrics such as lead time, cost or quality of results, which in turn support budgeting and resource allocation for the project.
- **Process execution and control** purposes, supporting organizations to monitor project progresses at any time and to apply corrective actions when it is needed. It is argued that process models allow management to compare the current state of the project with the desired state and this supports re-planning activities that identify strategies capable of steering the project towards a new desired direction.
- And **organizational development** purposes, supporting engineers to search for continuous improvement of processes, to capture and store knowledge arising from projects and to perform training sessions aiming to develop the skills and knowledge of the workforce about which activities to perform, tools to use or about how to overcome typical problems and risks.

There is thus a wide range of purposes for process models but all aim to provide some type of planning support. The following section explores the author's view about key challenges that modellers have to face and overcome to represent the dynamics of complex product development.

## 5.2 Challenges for modelling

The design and development *process* of large technical systems is, by itself, a *complex system*. Understanding and predicting the outcome of these design processes – its duration, cost, design solution quality, etc – is a difficult task for academia and industry. There are many examples of the unpredictability of these design processes. For instance, it is common that the design process of large technical systems lasts longer than planned, costs more than expected and delivers a solution with lower quality than anticipated. The events surrounding the entry-into-service of the Airbus A380 or the Boeing 787 Dreamliner aircrafts, where major unanticipated design and development issues delayed the programs and had huge financial costs, are examples of our current lack of understanding about the outcome of complex design processes. The underlying reasons for such phenomena arise from complexity and uncertainty generated by interactions between *social* and *technical* factors in these design processes. This section presents and discusses four key features which the author believes that need to be represented in models of complex design: uncertainty, iteration, collaboration and adaptive behaviour.

### 5.2.1 Uncertainty

The design process can be viewed as a “system of interrelated activities” (Wynn et al., 2011) which are performed with the purpose of reducing uncertainties surrounding the design solution or, in other words, as a system of design activities organized around the goal of gaining knowledge and confidence that the solution will function and perform according to the specified intent.

A distinguishable characteristic of complex design processes is the presence of many sources of uncertainty that need to be resolved through the interrelation of different kind of design activities. De Weck et al. (2007) mentioned both exogenous (such as changes in operational environments, in customer demands or in regulations) and endogenous (such as those arising from the maturation of technologies and corporate strategies) sources of uncertainty. Earl et al. (2005) identified lack of completeness, accuracy, consistency and quality in data (including experimental measurements) used by engineers during de-

sign activities together with ambiguity and lack of clarity in design descriptions as other sources of uncertainty. The review presented in Chapter 4 showed also that variability or aleatory uncertainty arising from uncontrolled factors, model uncertainty arising from approximations used in engineering models and decision-making uncertainty – defined as design imprecision – are sources of uncertainty affecting the prescription of values to variables that are vital to the definition of a design solution. In addition, other authors view also system complexity itself – such as interdependencies arising from system decomposition – as a source of uncertainty with impact in design processes (Wynn et al., 2011). Uncertainty is thus a key feature which needs to be represented in models of the design activity.

### 5.2.2 Iteration

The design process – viewed as a system of interrelated activities – normally consists of design activities performed in different modes. Pahl and Beitz (2007) explain that one mode consists of *analysis and synthesis*, which is based on search and discovery of new ideas and solutions or on combination and composition of existing ideas and sub-solutions to form a new whole. Activities performed in analysis and synthesis are thus about generating a novel design solution while the engineer is driven by a creative or imaginative force. Another mode of designing consists of *problem-solving* (Asimow, 1962), which relates essentially to the repetition of cycles of information processing followed by adjustments with the purpose of improving, refining and balancing the design solution that has been created or adjusting it to changes in input information, such as new or updated requirements.

Complex design processes, such as the preliminary design of gas turbines represented in Figure 4.3, are characterized by cycles of interrelated synthesis and problem-solving activities that involve multiple design teams. Section 4.4.1 described how system and sub-system design solutions evolved through iterative design loops involving communication of requirements and parameters across various design teams. The author’s empirical observation at Rolls-Royce showed also that design processes are often organized around

different repetitive patterns. These design loops or repetitive patterns are commonly known as *iteration*. An interesting and comprehensive perspective reported by Wynn et al. (2007) proposed several dimensions of iteration that can normally be found in complex design:

- **Exploration**, where iteration consists in a repetitive but rough analysis of a large spectrum of candidate design solutions.
- **Convergence**, defined as a iteration performed to converge upon one or a few satisfactory solutions, normally following prior iterative exploration.
- **Refinement**, viewed as iteration aiming to perform parameter trade-offs and to further improve design solution attributes.
- **Negotiation**, which refers to iteration occurring as a result of integration activities where goals and attributes need to be changed after negotiation taking place between design teams.
- **Rework**, where iteration is conducted to correct design problems or errors that have been discovered or to integrate new information arising from external sources to the design team, such as changes in requirements.
- **Repetition**, viewed as iteration consisting of the repetition of the same activities to achieve different goals or to refine different aspects of the design solution.

These dimensions of iteration proposed by Wynn et al. (2007) are illustrated in Figure 5.1 according to empirical observations made by the author at Rolls-Royce in the context of gas turbine design. Iteration is thus a second key feature which needs to be represented in models of design and relates to the continuous reduction of uncertainty levels.

### 5.2.3 Collaboration

Collaboration is viewed as a process where multiple people – with different intentions and backgrounds – work together to achieve a common and greater goal (Lu et al., 2000,

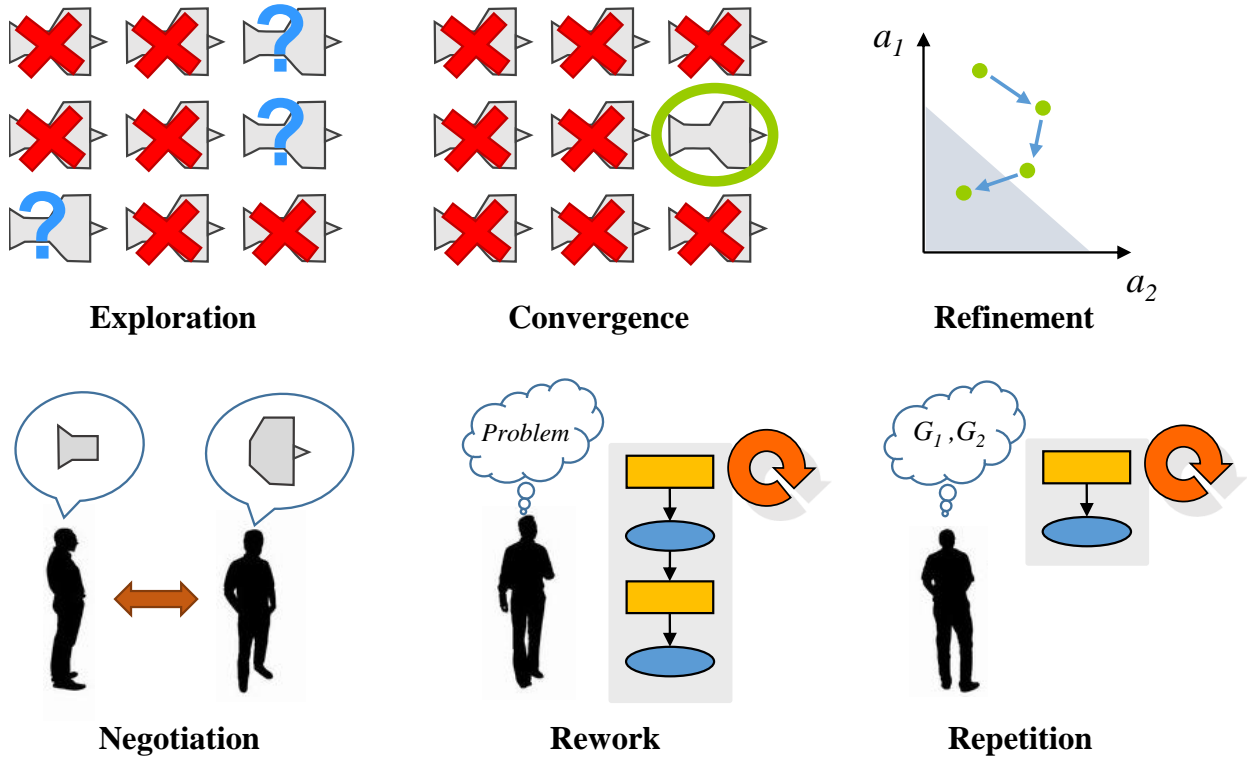


Figure 5.1: This thesis' view of the dimensions of design iteration proposed by Wynn et al. (2007) in the context of gas turbine design.

2007). Because of that, complex design has been recognized also as a “*social and collaborative process*” (Shai and Reich, 2004), where several aspects of collaborative behaviour have been found. For instance, Shai and Reich (2004) explain that collaboration is characterized by the search of mutual understanding about the terminology and the scope of work during the initial stages of design and development projects. A similar finding is referred by Movahed-Khah et al. (2010), who reported that interactions between design agents occurs with the purpose of achieving an “organized comprehension”, a common representation of the knowledge related to the design problem and a mutually accepted “lexicology, schema or language” in which to communicate during design activities.

In addition, in later stages of the project – when the problem has become clear and agreed upon – design work progresses through concurrent synthesis and problem-solving activities performed by different engineers or design teams (Shai and Reich, 2004). During this stage governed by *concurrent engineering*, a large amount of collaboration takes place through interactions between design teams and/or different domain experts intending to

resolve conflicting goals and trade-off portions of the design solution (Veeke et al., 2006). Collaborative behaviour during complex design thus also includes a key aspect of conflict resolution, where engineers work together to achieve a better overall design, i.e., towards a globally optimized solution which is not necessarily the sum of locally optimized sub-solutions (Case and Lu, 1996).

The author has reported also various examples of collaboration found in the context of gas turbine design at Rolls-Royce. Section 4.4.1 discussed that the preliminary design stage is largely governed by major collaborative design loops involving the system design team and the sub-system teams, which are composed of many engineers specialized in different components or specific engineering disciplines, such as aerothermal design or structural design. Figure 4.3 illustrated that these component designers and discipline specialists engage effectively into collaborative multi-discipline design iterations where they must come to agreements regarding the value assigned to design variables in a process leading to convergence towards an optimized solution at the different levels of the product's functional hierarchy.

#### **5.2.4 Adaptive behaviour**

Another distinguishable characteristic of complex design processes is the *adaptive* decision-making behaviour observed in the designers and engineers participating in such processes. Wynn et al. (2007) describe that design is different from repeatable processes which can be defined *a-priori*, since uncertainty and complexity forces engineers to accommodate evolving requirements, explore opportunities, master continuous changes, perform complicated trade-offs and solve design problems which always exhibit, to some extent, a certain degree of novelty. Lévérdy (2006) stresses also that complex system development is driven by continuous measurement and control of process activities and frequent decision points, where teams evaluate the actual state of the design and the project's key variables – such as time available, level of risk and cost – using measured data to make informed decisions about the next actions that should be taken to satisfy both internal and external stakeholders. Changing resources, negotiating milestones and/or re-planning activities

are examples of management interventions found often in practice by Wynn (2007) to adapt to dynamic environments and ensure that design problems are resolved or project delays are mitigated.

Complex design processes are thus adaptive and their outcome largely depends upon “in-situ decisions” made by designers and engineers (Wynn, 2007). The author found also examples of adaptive behaviour during the field research performed at Rolls-Royce. For instance, interviews to sub-system and component designers involved in the preliminary design stage represented in Figure 4.3 referred that it was often needed to choose and negotiate with project lead an appropriate level of activity fidelity, which was dependent upon the level of technical risk perceived by the project and the time available to deliver a solution to the customer (following the arrival of a Request-for-Information or a Request-for-Proposal). The level of fidelity was influenced by the use of 1-D, 2-D or 3-D design tools in aerodynamic or structural design activities. These allow engineers to achieve different levels of confidence about the accuracy of the results characterizing the behaviour of the design solution, at the expense of different activity durations and cost. Another example of adaptive behaviour found was the optional realization of some activities – such as off-design studies represented in Figure 4.3 – according to evaluations of risk and time available and subject to negotiation between designers and project lead.

### 5.3 Process modelling approaches

This section reviews and discusses the main modelling approaches that have been previously reported in product development literature, which include *activity-based models* and its variants, *agent-based models* and *system dynamics models*. The author found that there is a wide range of contributions surrounding these modelling approaches which have been discussed in major and recent reviews of literature on the topic made by Wynn (2007) and Browning and Ramasesh (2007). The former review analyzed more than 400 prior journal publications reported between 1994 and 2005 focusing *solely* on activity-based models. In virtue of the extent and quality of recent reviews, the following sections aim to provide only a comprehensive review of **selected** contributions which influenced the author’s

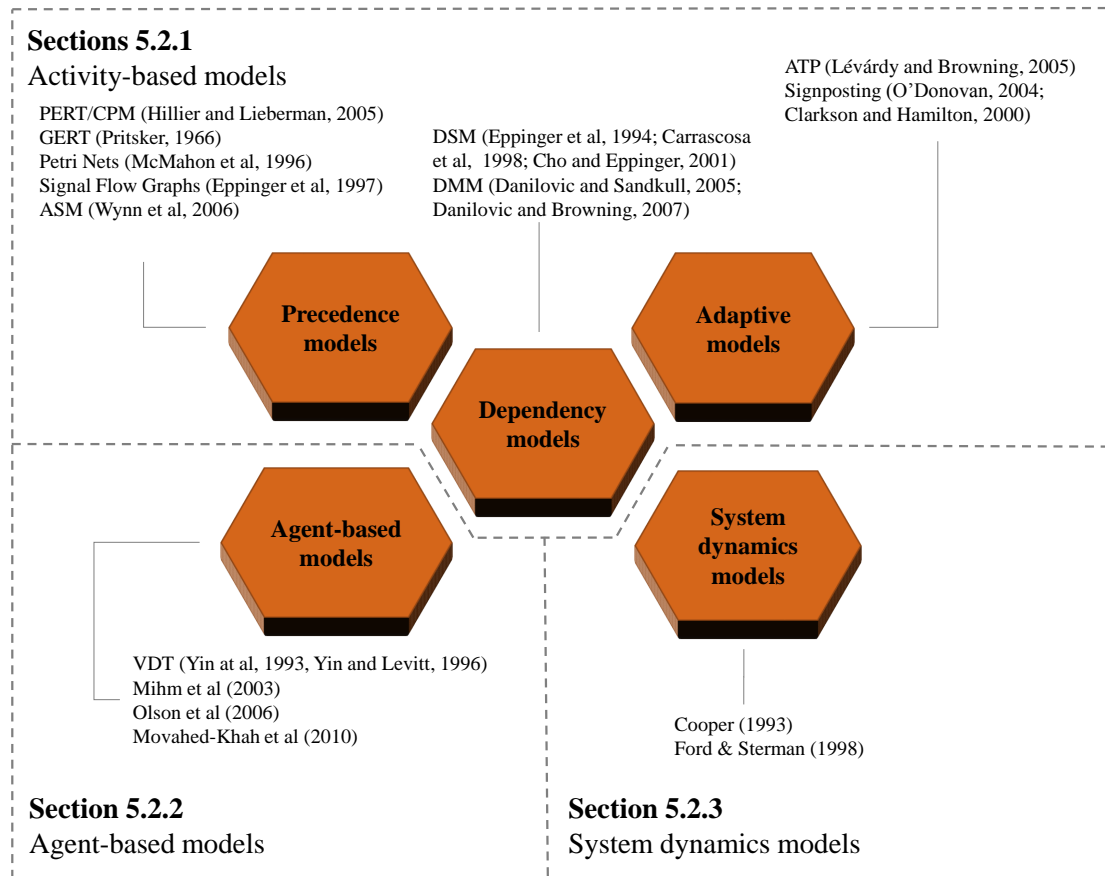


Figure 5.2: Overview of the literature review on product development modelling performed in this chapter.

research. Figure 5.2 presents an overview to the selected contributions surveyed in this dissertation. The chapter focuses subsequently in the discussion of the main strengths and limitations of each approach and its implications in the choice of modelling approaches to research the effects of requirements change in complex product development.

### 5.3.1 Activity-based models

Activity-based models view the design process as an “information processing system” (Browning and Ramasesh, 2007). This system is constituted by a *network* of independent *design activities* or *design tasks*<sup>4</sup>. The perspective followed in activity-based models is a “mechanistic” one (Wynn et al., 2007). This perspective assumes that the network of

<sup>4</sup>*Activity* and *task* are terms that have been used in literature to designate a distinct unit or element of design work. However, Wynn (2007) considers that the term activity should be used to refer to actual work while task should refer to a work element represented in a model. This thesis considers both terms are interchangeable in practice.



activities in the model represents the transformation of an initial set of input information characterizing the design problem into output information describing the behaviour of the design solution. Models based on information relationships of *precedence* and *dependency* have been found in literature together with models that attempt to incorporate a degree of *adaptation* during the activity selection. These three subtypes are analyzed in this subsection.

### 5.3.1.1 Precedence models

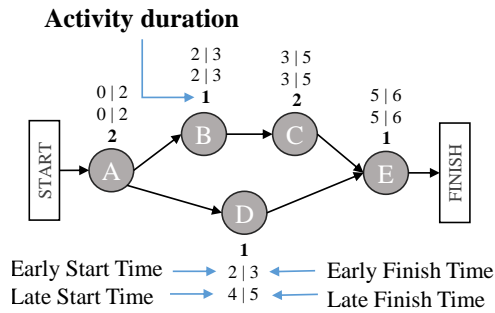
Precedence models represent the design process as a *pre-determined* network of activities. The structural relationship between activities relies on the concept of information precedence, which implies that the flow of information through the network attempts to represent the chronological order or temporal schedule followed by engineering in practice. Due to the strength of the precedence relationship, an activity planned to occur at a later point in the process flow is typically only allowed to begin after it has received all inputs from prior activities it relates to. Several precedence-based approaches have been investigated and applied to model product development – these approaches are illustrated in Figure 5.3 – relying on the following fundamental principles:

- The **Program Evaluation and Review Technique (PERT)** and the **Critical Path Method (CPM)** use a network diagram composed of *nodes* and directed *arcs* to model a process, as shown in Figure 5.3a. In recent formulations of PERT and CPM (Hillier and Lieberman, 2005), activities are typically represented on network nodes<sup>5</sup> while directed arcs capture the “right” sequence of activities that needs to be performed to achieve the project’s goals and the precedence relationships. It is assumed that activities are only attempted once and that the start of an activity can only occur after all related predecessors have finished. Based in this formulation, PERT uses a probabilistic approach that considers the best, worst and most likely duration of activities to estimate the expected process completion date (or the probability of completion before a certain date), while CPM determines the

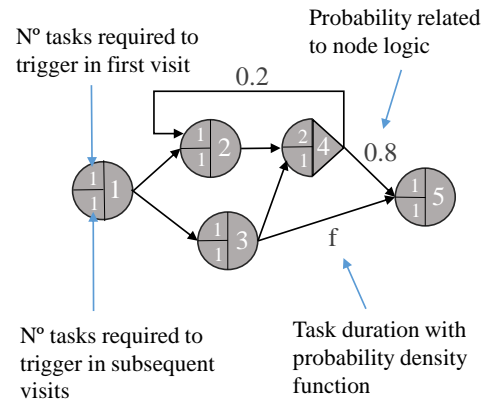
---

<sup>5</sup>Initial versions of PERT represented activities on arcs and milestones on nodes (Wiest, 1969).

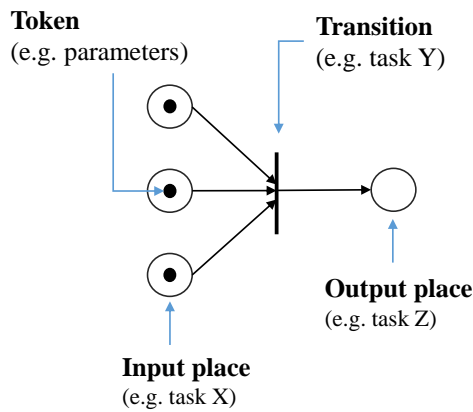
a) PERT/CPM



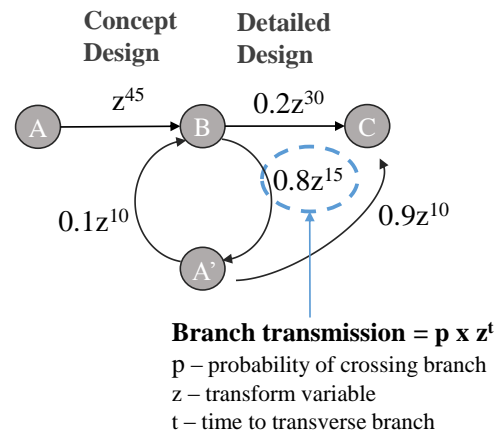
b) GERT



c) Petri Net



d) Signal Flow Graph



e) ASM

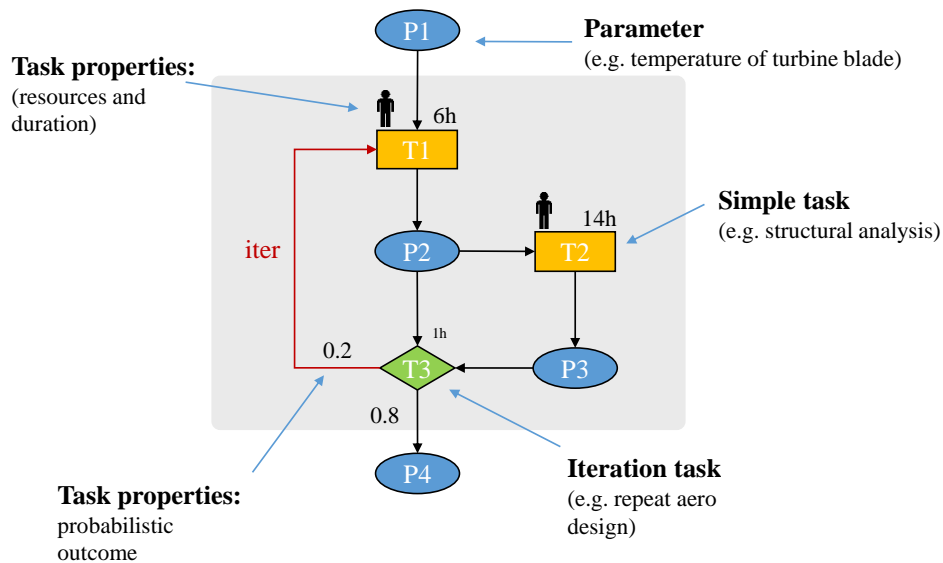


Figure 5.3: Overview of the activity-based models relying on precedence relationships: a) PERT/CPM; b) GERT; c) Petri Net; d) Signal Flow Graph; e) ASM.

calendar time required to complete the project through the analysis of the longest path of activities existing in the network.

- The **Graphical Evaluation and Review Technique (GERT)** introduced by Pritsker (1966) enables viewing the design process as a stochastic network. Activities were originally represented in *branches* (another term for directed arcs) and milestones on nodes (Pritsker, 1966). Through the introduction of additional notation, GERT allows the specification of the time required to cross branches as probability density functions – thus allowing stochastic activity durations – and the inclusion of logical relations in nodes also using probabilities (Figure 5.3b). This permits for instance to model that some activities shall not be performed in some circumstances or to model the repetition of activities, i.e. iteration, in other situations. In combination with Monte-Carlo techniques, GERT allows also a stochastic estimation of the total duration of the process or project represented by the network.
- The **Petri Net** is a logical network composed of two types of nodes: *places*, which are represented as circles, and *transitions* which are shown as vertical bars (Murata, 1989) as illustrated in Figure 5.3c. The arcs in a Petri Net connect places and transitions. In addition, a Petri Net contains a discrete number of markings – called *tokens* and shown as solid dots – which are assigned to places. Tokens can move between two places in the network when a transition separating the places is enabled and fires. Firing is allowed when the amount of tokens in input places matches the weight of the arcs connecting input and output places. Murata (1989) stated there is a wide range of applications of Petri Nets, such as in communication protocols, distributed software systems or manufacturing control systems. Within the context of product development, McMahon and Xianyi (1996) explored the potential of Petri Nets to model the design process of an automotive crankshaft and automate the interactions of concurrent activities that need to exchange information during design integration.
- The **Signal Flow Graph** is a network-based model formed by branches connect-

ing nodes where each individual directed branch has an associated quantity, called *branch transmission*. It has been used extensively in electrical engineering and to model discrete-event systems. Signal Flow Graphs is also based on precedence relationships between activities and Eppinger et al. (1997) has proposed the approach to model the design processes using the branch transmission to represent the probability of execution and the duration of the activity executed in the branch (Figure 5.3d). This mathematical representation allowed to determine through simulation the probability distribution of lead time of a simplified industrial die design process and the realization of sensitivity analysis to understand the impact of iteration (Eppinger et al., 1997). Isaksson et al. (2000) used also Signal Flow Graphs to compare the performance of process alternatives in aero engine design.

- The **Applied Signposting Model (ASM)** is a model-based approach also relying on information precedence which has been proposed by Wynn et al. (2006) to support both process visualization and planning through simulation. ASM provides a graphical framework allowing the construction of the process network based on key building blocks which have been integrated in a software package named as Cambridge Advanced Modeller (CAM, 2013). These building blocks include several types of network nodes represented in Figure 5.3e: *parameters*, which represent informational aspects of the design object; *simple tasks*, nodes with one input and one output; *iteration constructs*, decision nodes with one input and two output scenarios; and *compound tasks*, which have one input and any number of output scenarios. Arcs from task to parameters are defined as *data interactions* while those from parameters to tasks are known as *flow interactions* in ASM. Moreover, ASM allows for hierarchical modelling and the specification of more sophisticated node properties, such as the parameter scope, probabilistic task duration, task execution pre and postconditions and task resource requirements (Wynn et al., 2006). Some of these features are also depicted in Figure 5.3e. Monte-Carlo simulations in the Cambridge Advanced Modeller allow the stochastic investigation of process duration, task scheduling and scenario analysis.

### 5.3.1.2 Dependency models

Models based on relationships of information dependency are a second type of activity-based models found in prior literature. Dependency has been considered a weaker relationship than precedence (Wynn, 2007), since it does not imply a pre-determined ordering between interdependent model elements. Dependency approaches used in design process modelling thus incorporate the coupling of information between activities, but do not encode from the start how the sequence of activities should be executed. Two major dependency-based models can be found in prior product development literature:

- The **Design Structure Matrix (DSM)** proposed by Steward (1965) relies on the idea of storing dependencies or interactions between elements in a  $N^2$  matrix, where elements are positioned in rows and columns and element dependencies are identified through a mark in the corresponding matrix cell. Section 2.4 introduced the DSM as a tool used to perform requirements traceability analysis and which supports several change propagation methods. Browning (2001) identified four types of DSM in product development: the *Activity-Based* (or Schedule) DSM, to model the information flow between design process activities; the *Parameter-Based* DSM, to model design parameter interdependencies; the *Component-Based* (or Architecture) DSM, useful for modelling system component relationships and investigate architectural decomposition; and the *Team-Based* (or Organization) DSM, aiming to capture the interactions between people or organizational structures. In the design process context, (Eppinger et al., 1994) showed that the Activity-Based DSM supports the organization and re-sequencing of activities through the analysis of which activities can be performed in series or in parallel and which are coupled (requiring an information flow in both directions). In addition, several algorithms allow the analysis of structural relationships encoded in the DSM. For instance, *clustering* allows the identification of blocks of activities which are tightly coupled and *partitioning* algorithms transform any DSM into a lower-triangular form (to the extent possible), which allows the identification of feedback loops and the search of process improvements reducing rework of upstream activities (Dong, 2002). Fur-

thermore, Carrascosa et al. (1998) used a stochastic simulation DSM model based on the probability of activity change and the impact of that change on related activities (in terms of rework) to estimate the likelihood of completing the design process on time. Similar DSM-based simulation approaches have been also proposed to estimate the probability distribution of lead time, analyze scenarios and search for process improvements (Cho and Eppinger, 2001).

- The **Domain Mapping Matrix (DMM)** is an extension of the DSM concept also relying on information dependency. Danilovic and Sandkull (2005) coined the DMM as a  $N \times P$  matrix linking two DSMs, one of size  $N \times N$  and the other with a dimension of  $P \times P$ . The DMM was proposed to explore and understand dependencies across *multiple domains* of product development, such as different projects, processes and organizational structures (Danilovic and Sandkull, 2005) or to relate typical domains of interest in complex system development, namely the requirements domain to the architectural or functional domains and the latter to the design parameters' domain (Danilovic and Sandkull, 2005). The inter-domain DMM – DSM approach has been also baptized as the Multiple Domain Matrix (MDM) by Eichinger et al. (2006). A recent area of application of this concept has been in multidomain change propagation analysis – as mentioned in Section 2.4 – where several authors have attempted to use DSMs in combination with the DMM to model the propagation of changes in requirements to the product's functional domain (Hamraz et al., 2012; Koh et al., 2012).

DSM and DMM have achieved a remarkable popularity and a considerable number of publications with activity-based model applications can be found (Browning and Ramasesh, 2007). Some authors, such as Wynn et al. (2007), mention also *IDEF0* (NIST, 1993) as an additional dependency-based approach. IDEF0 is known as a function modelling approach where the concept of function is the main building block of models. A function allows the transformation of *inputs* into *outputs* under some *control* conditions and is supported on *mechanisms* that represent the resources used to execute the function. The large-scale review made by Browning and Ramasesh (2007) showed however that IDEF0 has not

been extensively explored in product development process modelling. Consequently, this dissertation shall focus mainly in the comparative analysis of DSM and DMM with other types of models.

### 5.3.1.3 Adaptive models

Adaptive models are the third type of activity-based models and rely in adaptive principles of task selection. Adaptive models view the design process as a dynamical process where the selection of tasks to be performed depends upon *state variables*. These state variables are normally related to the design process condition or replicate properties of the product or the decision-makers involved. The relationship between activities is thus no longer based on information precedence or dependency. Dynamical algorithms or schemes are normally encoded in adaptive models to guide activity selection. These algorithms ultimately define the process flow that arises from the model. Key adaptive models found in product development literature include:

- The **Signposting** model proposed by Clarkson and Hamilton (2000), which is based on an activity selection scheme that attempts to optimize the level of *confidence* in the design parameters used as inputs in the next activity at each time step of the simulation. This state variable is represented using three discrete levels: low, medium and high, which are respectively associated with an un-proven, feasible and satisfactory design solution. In Signposting, the execution of a selected activity causes the level of confidence of solution parameters to change in some manner and the process finishes once a certain level is achieved (Clarkson and Hamilton, 2000). The algorithm was initially tested and evaluated in a simplified representation of the helicopter rotor blades design process. Extended versions of Signposting were proposed subsequently incorporating stochastic activity duration, stochastic outcomes from activity execution and activity resource needs (ODonovan et al., 2004).
- The **Adaptive Test Process (ATP)** models projects in terms of their design activities introducing the concept of activity *modes* (Lévárdy and Browning, 2005; Lévárdy, 2006). Each mode corresponds to a particular “version” of the activity

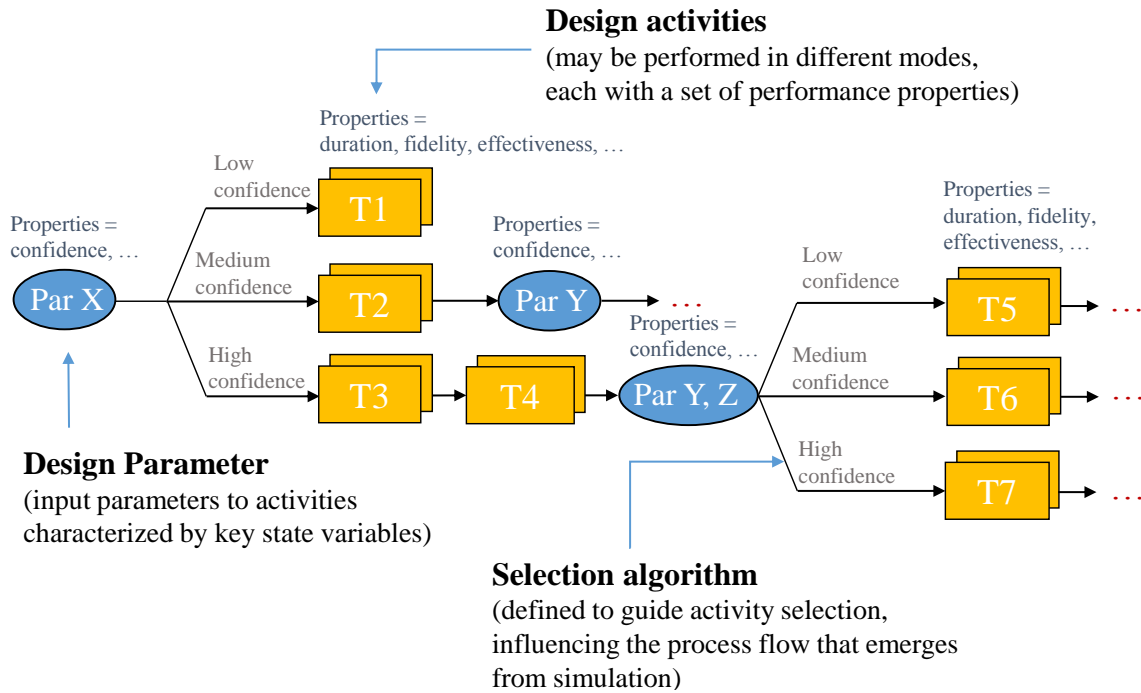


Figure 5.4: Overview of key features of activity-based adaptive models found in literature, such as Signposting Clarkson and Hamilton (2000) and the ATP (Lévárdy and Browning, 2005).

with different properties but intending to fulfill a similar goal. Lévárdy and Browning (2005) define the duration, cost, fidelity, effectiveness, availability and value as properties of each activity mode. Moreover, each mode has also different entry criteria to guide its selection and produces output results of different quality. The entry criteria is defined as the level of maturity of inputs to perform the activity in a certain mode, which is similar to the concept of maturity found in Signposting. The goal of the adaptive algorithm defined in ATP is to select the activity that adds more value to the process depending upon the state of the project, which is defined according to its cost, duration and risk profiles. Lévárdy (2006) explored the model through Monte Carlo simulations relying on stochastic sampling of all the parameters defined.

Figure 5.4 illustrates some of the key features of activity-based adaptive models discussed in this section from the author's analysis of Signposting and the ATP model.



### 5.3.2 Agent-based models

Agent-based models is a class of modelling approaches that views complex systems as an aggregation of autonomous and interacting agents (Macal and North, 2010). Figure 5.5 illustrates that a typical agent-based model has several key elements: various *agents*, with individual behaviours and decision-making rules; different agent-to-agent or *multi-agent interactions*, which influence individual behaviours and decision-making; and some *environment*, where agents “live” and which allows agent interactions to take place. This set of elements is common to most models. Figure 5.5 shows also that unlike other modelling approaches, complex systems are modelled using a *bottom-up approach* in agent-based models, which implies that the overall behaviour of the system is not explicitly modelled but *emerges* from the behaviours prescribed to the individual agents and from multi-agent interactions.

The range of applications of agent-based models is wide and spreads across various scientific domains, such as social sciences, natural sciences and computer science. In computer science, an agent has been defined as a software construct living in some environment that is capable of performing independent actions to achieve its objectives (Wooldridge and Jennings, 1995). Applications for software agents range from regular software program “services” or “daemons” to intelligent agents with learning capabilities developed for managing internet shopping services, complex air traffic control systems or advanced security and surveillance systems (Wooldridge, 2009). Advanced applications in computer science have been fostered by the growth of computer power and the need of delegating large tasks to distributed and concurrent systems and has led also to the development of multi-agent systems, which have been defined by Wooldridge (2009) as computer systems composed of many autonomous agents capable of cooperating, coordinating and negotiating tasks in order to solve a complex engineering problem.

Conversely, the goal of agent-based modelling application in social and natural sciences has been the study of complex social and natural systems and the generation of explanatory knowledge about the systems’ behaviour. Axelrod (1997), who applied agent-based modelling to study the complexity of cooperation using the Prisoner’s Dilemma, states that

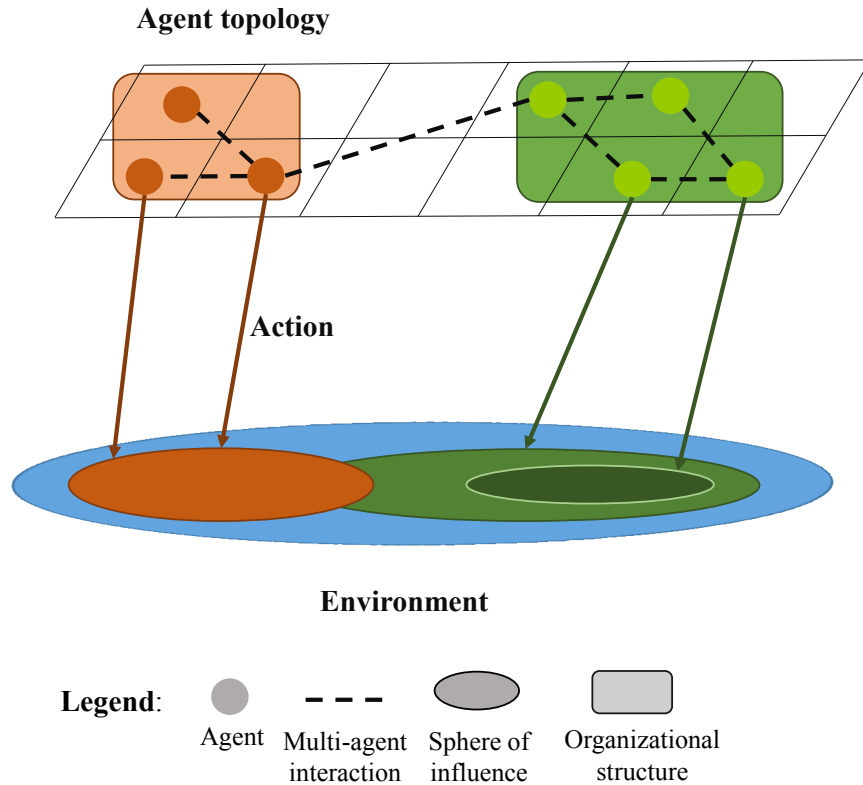


Figure 5.5: Overview of the key elements of agent-based models. Adapted from Macal and North (2010) and Wooldridge (2009).

the use of simulation relying on agent-based models provides a “way of doing experiments” and “generating data” from which emergent properties and patterns characterizing the system’s behaviour can be induced from reasoning<sup>6</sup>. Furthermore, Axelrod argues that bottom-up techniques such as agent-based models approach allow a scientific approach to the study of complex systems composed of many interacting actors where the “patterns of interaction are too difficult for a mathematical solution”. Due to its bottom-up nature, Macal and North (2010) states also that agent-based models are suited to assess the effects of *self-organization*, *learning* and *adaptation* in complex social or natural systems. In the scientific domain of this thesis, Section 5.2 has demonstrated that complex product development has been viewed also by many authors as a social and collaborative process where many interactions take place between design actors and where the adaptive behaviour of individual actors influences decisively the outcome of the whole design process.

<sup>6</sup>Inductive reasoning is used by Axelrod to refer to the search of conclusions from data patterns and is clearly distinguished from mathematical induction used during theorem proving.

Wynn et al. (2007) coined this view of product development – which emphasizes the importance of social interactions, cooperation and in-situ decision-making about the next actions to take according to the perceived state of affairs – as *actor-based*. Recognizing the importance of this perspective, several key examples of agent-based models in product development can be found in prior literature:

- The **Virtual Design Team (VDT)** was an agent-based model developed by Jin et al. (1993) to support the design of organizational structures in complex projects. The model represented designers and design teams as agents with specific tasks, goals, allocation capabilities, communicational intensities and coordination behaviours within a network representing the organization executing the project. The VDT platform relied on a object-oriented and discrete-event computational implementation (Jin and Levitt, 1996). The model could be used, for instance, to investigate the performance of the projects considering the introduction of new communicational tools to agents (e.g. video conferencing) (Jin et al., 1993). Subsequent versions of the VDT platform incorporated measures of activity flexibility, complexity, uncertainty and interdependence strength and were used by Levitt et al. (1999) to investigate how alternative organizational changes affect the performance of satellite design projects.
- **Mihm et al. (2003)** proposed an agent-based model to investigate the causes of oscillations in product development leading to project performance deterioration. Their model assumes that a change of a component design by an agent leads to an update of components being designed by other agents and there is a limited amount of information exchange between agents. Based in this model, Mihm et al. suggest that design oscillations become more frequent when systems have a higher number of structural interdependencies and discuss management strategies to mitigate oscillations, such as product modularization and the release of preliminary information.
- **Olsen et al. (2009)** developed an agent-based computational platform to inves-

tigate how organizational interdependencies and a team's resources and knowledge affects the dynamics and the emergent system-level properties of organizations during the execution of design projects. Unlike the VDT approach – who sought a general purpose model – the work of Olsen et al. presented a multi-agent model and simulation implementation focusing in the conceptual design of space missions. The model was largely based on empirical observations made by the authors about a complex design process executed by NASA and attempted to include the dynamics observed, namely the distribution of social and task constituents, procedural processes executed by system and sub-system design teams, information concurrency and collaborative design loops including negotiation of key sub-system design variables. The approach ultimately aimed to demonstrate the effectiveness of a particular organizational design to achieve the project's goals.

- The **Intelligent System for Interaction Analysis in Design (ISIAD)** is an agent-based computational system developed by Movahed-Khah et al. (2010) to perform interaction analysis of collaborative and distributed design processes. The model's goal was to support the discovery of interaction structures and patterns – such as complete, bilateral and quasi-null cooperation (Movahed-Khah et al., 2010) – occurring between designers and design teams during collaborative and distributed design and to support the investigation of the dynamics of self-organization.

### 5.3.3 System dynamics models

In addition to activity-based and agent-based models, system dynamics models offer a third distinct view of complex design processes. System dynamics models decompose complex systems into a set of key *stock* and *flow* elements (Sterman, 2002). Stocks are elements which *accumulate* or *consume* some system quantity over time while flow elements define the *rate of change* of stocks. Causal diagrams defining the system's elements, their interactions and the main reinforcing and balancing feedback loops are typically used to represent the dynamics that the model intends to study. Models require also the prescription of mathematical equations governing the flow of quantities through stocks. The

overall dynamic behaviour of the system arises from simulations solving these equations. A couple of implementations of system dynamics models are known in product development literature. Cooper (1993) presented a revised model of the “rework cycle” based on a system dynamics model. The model captured the amount of undiscovered rework – work reported as done but containing errors that are discovered later – as a function of the work to be done, the resources available, their productivity and the quality of outputs. Results from simulation led Cooper to argue that undiscovered rework is the “single most important source of project cost and schedule risk”, since rework discovered by downstream design effort requires additional staff for a time longer than initially expected. In another research contribution, Ford and Sterman (1998) described a multi-phase project model of semiconductor chip development accounting for processes and resources also based on a system dynamics model. Each project phase contained a stock model including the tasks to be iterated, completed, approved and released (Ford and Sterman, 1998). The corresponding flows governing the cascade process (the rates of iteration, completion, approval and release) were modelled as functions of the resources available and task duration. Ford and Sterman argue that the model can be used to investigate the dynamic effects of processes and resources in project performance and improves planning capabilities compared with traditional static approaches, such as PERT.

### 5.3.4 Discussion

The identification and analysis of the previous approaches to model product development allowed to foster the author’s understanding about the main strengths and limitations of each type of approach. This section presents a critical discussion based on a comparative analysis of the capabilities observed in each approach to incorporate models of *uncertainty*, *iteration*, *collaboration* and *adaptive behaviour*. These features were identified in Section 5.2 as four dimensions observed during complex product development practice which this thesis argues that need to be represented in models due to their influence in the outcome of development processes.

The previous literature review showed that the different sources and types of uncertainty

discussed in Section 5.2.1 are normally represented in the different types of models surveyed through probability distributions of properties assigned to model elements. Examples include probabilistic treatment of activity durations, confidence of input variables or availability of resources assigned to node or arcs in activity-based models, such as GERT, ASM, Signposting or ATP. Similar assignment of probabilistic properties to matrix cell properties in DSM has been performed to represent activity durations and the probability or impact of a change affecting dependent elements in the activity network. A similar treatment of uncertainty occurs also in agent-based or systems dynamics models. For instance, the VDT approach modelled uncertainty in the communication between agents and in the quality of the work arising from agent activities also through probabilistic laws (Jin and Levitt, 1996). The main strengths of this common approach to uncertainty modelling is its simplicity and the ability to understand the effect of individual uncertainties on project performance through the analysis of results arising from Monte Carlo simulations. Limitations of the approach include the practical difficulty of eliciting from experts a large quantity of information characterizing the probabilities surrounding these properties, which impacts the verification and validation of models.

The ability of capturing the dimensions of interaction is however different among the modelling approaches reviewed. Activity-based models relying on precedence relationships – like PERT, GERT, Petri Nets, Signal Flow Graphs or ASM – depart from a pre-determined network of activities which facilitates the identification and visualization of iterations but effectively limits its application mainly to stages of the design process, such as detailed design stage, where the sequence of tasks and the patterns of iteration are well-defined. Recalling the six dimensions of iteration discussed in Section 5.2.2, this dissertation considers precedence-based models suited to model refinement and repetition, but with limited potential to capture the dynamics of exploration, convergence, negotiation and rework. For instance, the rigidity of the network-based model structure inhibits dynamic task selection, which is required to model exploratory iteration, convergence and rework, and prevents the introduction of frequent interactions between design teams which are required to capture iterative cycles with negotiation. Similar limitations appear also

in system dynamics models, due to the well-defined structures of stock and flow elements defined in models.

Conversely, activity-based models relying on dependency relationships – such as DSM – and adaptive principles – like Signposting and ATP – introduce additional flexibility in the definition of the process flow which augments their capability of capturing the different dimensions of iteration. Furthermore, the distributed nature and the flexibility in individual decision-making behaviour found in agent-based models appear as strengths allowing a more complete representation of the dimensions of iteration typically found during the early stages of complex design, where many cycles of iteration involve exploration, convergence and negotiation between multiple design teams. These models are however much less cost-effective to capture well-defined activity sequences and iteration cycles of refinement and repetition which are common during the detailed design stages of complex development projects. The author thus views precedence-based models and agent-based models in opposite ends of cost-effectiveness to model the dimensions of iteration typical of preliminary design and detailed design.

Collaboration is another dimension of complex product development addressed differently among the models surveyed. Modelling collaboration requires two key capabilities: a representation of the social network found in organizations – which is often hierarchical – where designers and teams are integrated; and the ability to allow frequent, concurrent and even asynchronous information exchange across that social structure of relationships. The absence of this social dimension in precedence-based models and the difficulty of incorporating concurrent process flows that communicate regularly are limitations which constrain this subgroup of models to account for the role of collaboration in complex development projects. Similar conclusions can be retrieved from the analysis of system dynamics models.

On the other hand, it appears possible – at least to some extent – to model concurrency and to connect the activity domain (or other technical domains) with the organizational domain using the DSM – DMM approach. Dependency-based models thus seem to contain a higher potential for collaboration modelling than precedence-based models. Relatively

to agent-based models, the bottom-up approach used in the models allows the social dimension of product development to be explicitly incorporated since agents can represent individual designers or teams with specific goals which are allowed to engage into multi-agent interactions to fulfill a higher-level objective. The disposition of agent-based models to represent social interactions and concurrent and distributed decision-making is intrinsic, which facilitates the inclusion of collaboration models. This thesis views this capability as a major strength of this type of model-based approach.

The ability to code adaptive behaviour in the agents' internal decision functions or rules is another strength of agent-based models, considering the fourth key characteristic of complex product development previously discussed in Section 5.2. The agents' ability to continuously "sense" the state of the environment and request information about the internal system's state through multi-agent interactions allows frequent and "in-situ" decision-making about its next actions. This suggests these models hold the potential to incorporate adaptive behaviours similar to the ones observed in human actors during complex system development, who frequently evaluate the actual state of the design solution against the time available and project risk/ cost in order to decide upon the next design action. This adaptive and "in-situ" decision-making capability can also be found in activity-centric adaptive models and, to some extent, also in dependency-based models that incorporate advanced simulation algorithms (e.g. Cho and Eppinger (2001)). Precedence-based models have strong limitations in this fourth aspect, since the process flow is largely pre-determined.

Capturing uncertainty, iteration, collaboration and adaptive behaviour is surely important for research. However, one must recall that the purpose of process modelling is to support visualization, process planning, control and execution and organizational development in practice, as discussed in Section 5.1. The cost-effectiveness of industrial implementation thus also becomes an important dimension for discussion. In this regard, precedence and dependency-based models are currently the most cost competitive approaches, since they rely upon principles and techniques simple to understand and a user-friendly graphical notation. The previous characteristics ease the user's implementation for planning and



control purposes and facilitates the use of these model types for group visualization and organizational development, such as staff training.

Its cost-effectiveness is a key strength and explains also why this type of models are used more often in industrial practice and have attracted most of the research attention during the last couple of decades (Browning and Ramasesh, 2007). In comparison, the lower ease of implementation and cost-effectiveness of both activity-based adaptive models and agent-based models are barriers that have limited so far their penetration in industrial practice. The cost advantage of precedence and dependency-based models is valid in small-to-medium model sizes, but reduces or even disappears in large models containing, respectively, hundreds/ thousands of nodes and similar number of matrix rows/columns. For instance, Bell et al. (2007) report a ASM model – representing a turbine blade design process – with 1300 nodes, 400 activities, 50 sub-processes and 11 levels of depth which exemplifies, from the author’s perspective, the relative loss of cost advantages for visualization, planning and control purposes as these models grow in size. Table 5.1 summarizes this section’s discussion around the key strengths and limitations of each model-based approach.

## 5.4 Review of progresses

This chapter provided a review of selected literature with the intent of addressing the first of the third group of research questions – Q3-a – identified in Section 1.4. Based on this review and on the previous critical analysis and discussion, the author elaborates below an answer to this research question:

**Q3-a** *What approaches are available to model complex and uncertain design processes?*

*What are their main strengths and limitations?*

Literature showed there are three main types of modelling approaches. The first type is designated as activity-based, which views complex design processes as a network of inter-related activities capable of transforming problem inputs into a design solution output. The nature of the relationship specified between activities has fostered the development of

Table 5.1: Summary of strengths and limitations of product development process modelling approaches

Approach	Strengths	Limitations
Activity-based, precedence	<ul style="list-style-type: none"> <li>• User-friendly, cost-effective</li> <li>• Provides good visualization, planning and control support in well-defined processes</li> <li>• Models well refinement and repetition iteration</li> </ul>	<ul style="list-style-type: none"> <li>• Difficult to model exploration, rework and negotiation</li> <li>• Difficult to account for the role of collaboration</li> <li>• Difficult to account for adaptive behaviour</li> <li>• Difficult to use large models, keeping cost-effectiveness</li> </ul>
Activity-based, dependency	<ul style="list-style-type: none"> <li>• User-friendly, cost-effective</li> <li>• Provides good visualization, planning and control support</li> <li>• Dependency allows to capture more types of iteration</li> <li>• Multi-domain models allow linking with social dimensions</li> </ul>	<ul style="list-style-type: none"> <li>• Difficult to account for negotiation and collaboration</li> <li>• Difficult to account for adaptive behaviour</li> <li>• Difficult to use large models, keeping cost-effectiveness</li> </ul>
Activity-based, adaptive	<ul style="list-style-type: none"> <li>• Allows to capture exploration, convergence and rework</li> <li>• Allows to capture adaptive decision-making behaviour</li> </ul>	<ul style="list-style-type: none"> <li>• Not user-friendly</li> <li>• Not cost-effective for well-defined processes</li> <li>• Difficult to visualize and implement for planning and control purposes</li> </ul>
Agent-based	<ul style="list-style-type: none"> <li>• Allows to capture exploration, convergence, rework and negotiation</li> <li>• Allows to capture social interactions and collaboration</li> <li>• Allows to capture distributed decision-making and adaptive behaviour</li> </ul>	<ul style="list-style-type: none"> <li>• Not user-friendly</li> <li>• Not cost-effective for well-defined processes</li> <li>• Difficult to visualize and implement for planning and control purposes</li> </ul>
System dynamics	<ul style="list-style-type: none"> <li>• Allows to account for dynamic effects arising from feedback</li> <li>• Provides planning and control support in well-defined processes</li> </ul>	<ul style="list-style-type: none"> <li>• Difficult to account for various facets of iteration</li> <li>• Difficult to account for collaborative behaviour</li> <li>• Difficult to account for adaptive behaviour</li> </ul>

three subtypes of activity-based models: precedence-based models, where the information flow between activities is pre-determined and rigid; dependency-based models, where activity couplings are identified but the order of execution is not rigid; and adaptive models, where activity selection is dynamic and depends upon state variables. The second type is the agent-based approach, which sees complex product development as a distributed system consisting of different actors, each with individual goals and behaviours, that are capable of interacting and coordinating actions to achieve a higher-end purpose. And the third type of approach is known as system dynamics, where the model incorporates stock and flow elements that regulate the rate of work or information flow.

The key strength of both precedence and dependency-based models is its cost-effectiveness to model moderate size and well-structured development processes. Because of that and their intuitive graphical notation, they have been applied in industrial practice and support well process visualization, planning and execution monitoring and control. Its limitations include the difficulty to include various forms of iteration and to account for collaboration and adaptive behaviours often observed during development projects. On the other hand, the key strengths of activity-based models relying on adaptive algorithms and agent-based models is precisely their capability to incorporate the different facets of iteration and models of collaboration and adaptive behaviour. This comes however at the expense of loss of visualization capabilities and cost-effectiveness to support planning and control, particularly for well-defined stages of the development process. The ability to capture high-level dynamic effects arising from feedback loops is the key strength of system dynamics models. However, this type of approach is also limited by the difficulty to account for multiple aspects of iteration, collaboration and adaptive decision-making.

## **5.5 Summary**

Recapitulating, the current chapter arose from the need detected during exploratory research of understanding and quantifying the effects of requirements change in complex product development projects. Departing from the argument that model-based simulation provides a cost-effective approach for doing virtual experimentation and to research

cause-effect relationships in complex systems, this chapter surveyed models of product development presented in prior literature. It was reported that the central purpose of models is to provide planning support to organizations, since complex systems development requires a multitude of plans to be created, used and managed during projects. The literature review showed that models of product development can specifically support process visualization, process planning, execution and control and support organizational development.

In addition, the key challenges that modellers have to face and overcome to represent the dynamics of complex product development were identified from literature and examples drawn from empirical observations made by the author during the time spent on-site at Rolls-Royce investigating projects and interviewing engineers and designers. The chapter argued that there are four key dimensions that need to be captured in models due to their importance to the outcome of these processes: uncertainty, iteration, collaboration and adaptive behaviour. The main modelling approaches that have been previously reported in literature – which include activity-based models and its variants, agent-based models and system dynamics models – were then presented and discussed based on a comparative analysis of their capabilities to represent the previous four dimensions.

The chapter concluded that the main strength of activity-based models relying on precedence and dependency relationships among activities is their ability to represent in a cost-effective manner well-structured processes, such as those typical of the detailed design project stages. Conversely, it was argued that activity-based models relying on adaptive algorithms and agent-based models are best equipped to represent ill-defined processes where there are concurrent, frequent and collaborative interactions between participants and the adaptive behaviour of design teams is often observed. Such characteristics are typical of the concept and preliminary design stages of complex development projects. This chapter showed also that the strengths of adaptive models and agent-based models come at the expense of visualization and cost-effectiveness to support planning, execution and control activities and these limitations have so far limited their introduction in industrial practices.

# Chapter 6

## An agent model of early design

Departing from the previous analysis of the strengths and limitations of different modelling approaches, this Chapter draws from the author's empirical observations of complex product development projects at Rolls-Royce and explores a model-based approach aiming to support planning taking into account expectable levels of requirements change arising during early design stages. Change is more frequent during early design and exploratory research reported in Chapter 1 revealed that there is lack of planning tools supporting engineers during these project stages, as well as lack of understanding about the effects of change in project performance.

This Chapter is organized around six main sections which present and explore the model proposed of early design. It begins by a presentation of the research objectives driving model development. Secondly, the choice of modelling approach is discussed together with the methodology followed during model development. Thirdly, details about the design and implementation of the model are presented. Fourthly, results arising from an initial set of exploratory simulations are presented and evaluated. The chapter ends with a discussion of the research progresses made and a summary of conclusions arising from the research.

## **6.1 Objectives**

The key purpose of this Chapter is to bring forward a model developed by the author capturing the dynamics of complex product development during early design stages (Q3-b) and to support the estimation of early project performance taking into account the level of expectable change (Q3-c). In addition, the ultimate aim of this Chapter is to foster the development of novel approaches supporting organizations planning projects during early design (Q3). The development of novel models and tools is important because this period – normally encompassing concept and preliminary design – exhibits particularly complicated dynamics which make planning a difficult task.

Model development efforts reported in this Chapter were inspired by the empirical study of the preliminary design process performed by the author at Rolls-Royce. Section 4.4.1 described that this stage is driven by regular requests of an engine solution sent from the customer to alternative suppliers of power systems, who need to respond with a design solution before a specified deadline. Updated requirements are captured by the supplier from these customer requests and include high level requirements such as thrust, specific fuel consumption, weight, unit cost or reliability. Once a request for information or a request for a proposal has been received, the system design team performs the design of the thermodynamic cycle at a chosen operational point and decides upon the general mechanical arrangement of the turbomachinery capable of generating the thrust required within the fuel consumption, weight and cost allowed.

Figure 4.3 illustrated that the functional requirements derived from the systems design solution are then communicated to all sub-system design teams, who are requested to perform their aerodynamic and mechanical preliminary design activities and respond with bids of key sub-system parameters, like efficiency, weight and cost. These parameters had been assumed by system designers during their design activities. Because of that, collaborative design iterations occur until a solution is reached that satisfies all design teams and optimizes the customers requirements or until there is no more time available to respond to the customer.

The study of the preliminary design dynamics allowed the author to comprehend that

early design normally involves a great deal of uncertainty, collaborative iteration and adaptive organizational behaviour. For instance, customer requirements are uncertain and change regularly, triggering new design iterations. The design solution evolves through changes in requirements and updates in solution parameters exchanged between system and sub-system teams during collaborative design loops. Multi-discipline design iterations involving aerodynamic and mechanical design are another example of collaboration also observed during component design to achieve sub-system convergence. And examples of adaptive behaviour observed by the author – also described in Section 5.2.4 – include the ability to request additional resources or to negotiate with design lead an appropriate level of activity fidelity, depending upon the time available and the level of perceived risk.

## 6.2 Research approach

Drawing from the review of literature and the analysis of the strengths and limitations of each modelling approach presented in Chapter 5, this section begins with a discussion supporting the author’s view about which modelling approach is appropriate to capture the dynamics of early design stages of complex product development. This stage is when requirements are changing often and thus it is when the effects of change most influence the progress of projects.

### 6.2.1 Approach selection

It is logical to conclude from the analysis of Table 5.1 that activity-based models relying on precedence and dependency relationships struggle to capture the dynamics typical of the early design stages of complex product development due to several reasons. Firstly, these models rely on a pre-determined and rigid activity network while the early design process flow is *loosely defined* around iterative cycles of exploration, convergence and refinement involving multiple stakeholders, teams and design disciplines. Secondly, key early design characteristics like concurrent and frequent interactions between teams – the work of different teams is represented by different sub-processes or process “lanes” in

activity-based models – are difficult to incorporate in simulation models without adding a large number of decision nodes. Models with high number of activity and decision nodes quickly become *intractable*, both to use and to communicate.

Moreover, activity-based simulation models normally rely on centralized discrete-event engines which are not suited to handle random or unscheduled events during simulation. This prevents capturing the dynamics of asynchronous information exchange between teams which, from the author’s point of view, is a key factor driving the distributed and collaborative process flow during early design. Fourthly, adaptive behaviour typical of early stages is also difficult to incorporate since, as previously referred in Section 5.3, these models rely on a mechanistic view of design projects.

Conversely, Table 5.1 showed that agent-based models and activity-based models relying on adaptive principles are better equipped to capture various dimensions of iteration, collaboration and adaptive behaviour which can be observed during early design stages of complex product development. Comparing the two alternatives, this dissertation argues that agent-based modelling is a more promising approach since it offers three major benefits relatively to adaptive activity-based models:

- **Organizational representativeness**, since they are built upon independent entities which can represent individuals or teams and can replicate the social structure observed in organizations in a detailed and intuitive manner. Organizational structures are absent or are represented less intuitively in adaptive models.
- **Social interactiveness**, since models allow individual agents to interact and communicate through message exchange which is mediated by an agent platform <sup>7</sup>. This social interactiveness facilitates the representation of collaborative processes and is absent in adaptive models such as Signposting or ATP.
- And **decentralized control**, since the global behaviour arises as a result of interactions between individual entities that have their own local control mechanisms.

Signposting or ATP appear to be governed by a centralized algorithm or set of

---

<sup>7</sup>The platform provides a virtual environment where agents “live” in. Further details about agent platforms are provided later in this Chapter.



rules driving the process flow. This centralization increases the difficulty of model development and limits the scalability of adaptive models.

This comparative analysis led to the adoption of **agent-based modelling** as the research method to tackle the objectives described above: capturing the dynamics of early design, investigating the effects of requirements change and developing novel approaches providing planning support.

### 6.2.2 Methodology

Based on the previous choice, this section describes the typical steps required to model complex systems using an *agent-based approach*. The concept of agent was introduced in Section 5.3.2 as a software construct situated in some virtual environment and capable of acting to meet its objectives in an independent manner.

In practical terms, an agent is a *software program*. Because of that, agent-based modelling can be made using any programming language. Recalling that different programming languages have also different purposes and capabilities, Macal and North (2010) state that agent-based modelling can be typically performed using: object-oriented languages, such as C++, Java or Python; or agent-based modelling dedicated software or toolkits, such as NetLogo (NetLogo, 1999), Repast Symphony (Repast, 2006) or AnyLogic (AnyLogic, 2013). The latter rely also in general purpose object-oriented languages like C++ or JAVA, but already offer an Integrated Development Environment (IDE) with pre-defined agent capabilities for accelerating model creation. Therefore, agents are normally developed using object-oriented programming, either directly or indirectly.

Object-oriented programming allows the creation of *objects*, which are particular computational instances belonging to a *class*. A class is a general template defining common *attributes* – e.g. data structures – and *methods* shared by all objects belonging to the class. Methods define all sorts of “actions” or “operations” that an object is allowed to perform, such as data processing or communicating to other objects. This permits the development of software as an interaction between different types of objects. Furthermore, object-oriented programming languages are built upon several fundamental properties:

encapsulation, which allows objects to manipulate components that are hidden and restricted from the external environment; inheritance, which allows objects to inherit all attributes and methods from a “parent” class; and polymorphism, which allows objects of different classes to implement variations of the same methods.

Since it is built upon objects, a key feature of an agent is thus that it is an *encapsulated* entity. It is thus a self-contained software program with well-defined boundaries and interfaces. Agents are embedded in a larger software environment over which they have partial control. They can receive information from this environment and make changes through actions they take. A distinctive feature relatively to objects is that agents embody a “stronger notion of autonomy” and independent threads<sup>8</sup> of control (Wooldridge, 2009). Wooldridge states that agents have control over their internal state and behaviours like objects, but are self-directed to achieve their goals and have more abstract reactive, proactive and social interaction behaviours than standard objects.

High-level social interaction behaviours are another distinctive aspect relatively to regular objects. Agents normally need to interact with each other to achieve their goals. Interactions are designed to be flexible, allowing agents to decide whether or not to engage with others according to an internal and “in-situ” evaluation of the context. Several communication languages have been developed to standardize how agents can interact and communicate within models. The KQML<sup>9</sup> (Finin et al., 1993) and FIPA-ACL<sup>10</sup> (FIPA, 1999) are two examples of agent communication languages which defined a standard format for passing messages between agents. Both languages define agent messages as objects belonging to a certain class of *communication acts* or *performatives*. FIPA-ACL includes 22 performatives, which allow messages of “request”, “call for proposals”, “propose”, “query”, “inform”, “agree”, “reject proposal” and “not understood” (FIPA, 1999), among others.

Macal and North (2011) also recently discussed the key steps required for modelling a

---

<sup>8</sup>In computer science, threads are separate streams of execution within a process that are handled independently by the operating system.

<sup>9</sup>The acronym stands for Knowledge Query Manipulation Language.

<sup>10</sup>The FIPA acronym stands for Foundation for Intelligent Physical Agents, while ACL stands for Agent Communication Language.

system using an agent-based approach. Their work allowed the author to systematize the following methodology for model development:

- **Intent definition**, specifying what is the system or problem, what information the model should provide and what are the main input and output variables.
- **Agent definition**, deciding upon how the system or problem's domain is naturally decomposed into or mapped against a finite number of distributed agents, where each represents a specific domain entity.
- **Environment definition**, deciding how to represent the environment each agent is allowed to see and interact with, as well as the type of environment. Several types of environment have been coined by Russel and Norvig: accessible or inaccessible for different agents; deterministic or non-deterministic, depending upon the treatment of uncertainty; static, if it changes only due to agent actions, or dynamic, if it changes in ways beyond the agent's control (Russel and Norvig, 1995).
- **Architecture and behaviour definition**, specifying the agent's internal structure and which actions each agent is allowed to perform on the portion of the environment it can interact, as well as the decision-making behaviour or decision rules for selecting each of the actions. This includes also interaction definition, i.e., defining how agents interact with each other using a particular communication language or interaction protocol and with the environment.
- **Platform definition**, selecting the most appropriate software platform according to the kind of agent-oriented programming required for model development.

This logical sequence is naturally iterative and resembles a waterfall model. It was adopted and followed by the author during the development of an agent-based model for planning and research of early design, which is presented subsequently.

## 6.3 Model development

The following subsections present and explore the **Agent Model for Planning and rEsearch of eaRly dEsign (AMPERE)**, which was developed by the author to address the research objectives described in Section 6.1. Discussion begins with an overview of the vision driving model development and of high-level input and output model variables.

### 6.3.1 Vision

Figure 6.1 illustrates that the AMPERE aims to include representations of organizational structures, design process activities, the effects of uncertainty, iteration, design collaboration within design organizations and adaptive decision-making behaviour and integrate them within the modelling approach. This dissertation argues that such integration constitutes a long-term *vision* for research in product development modelling that arises from the review of literature explored in Chapter 5. This review allowed the author to conclude that agent-based models could be used to develop and integrate more comprehensive representations of the different facets of complex product development.

In addition, the development of this agent-based approach was also triggered by this thesis' aim of investigating the effects of requirements change in project performance and support planning during early design stages. Figure 6.1 illustrates the main input and output variables driving model development. The typical level of change in requirements which should be expected in development projects are the inputs to the AMPERE. This level of change is defined through the metrics proposed and studied in Chapter 4: the Time between Changes (TbC) and the Magnitude of Change (MoC). In order to evaluate the effects of change in project performance, development cost and solution quality are defined as the outputs or dependent variables delivered by the AMPERE (Figure 6.1).

### 6.3.2 Development platform selection

The selection of a development platform for AMPERE followed a process of benchmarking. Two main criteria were used for the benchmark of alternative agent development

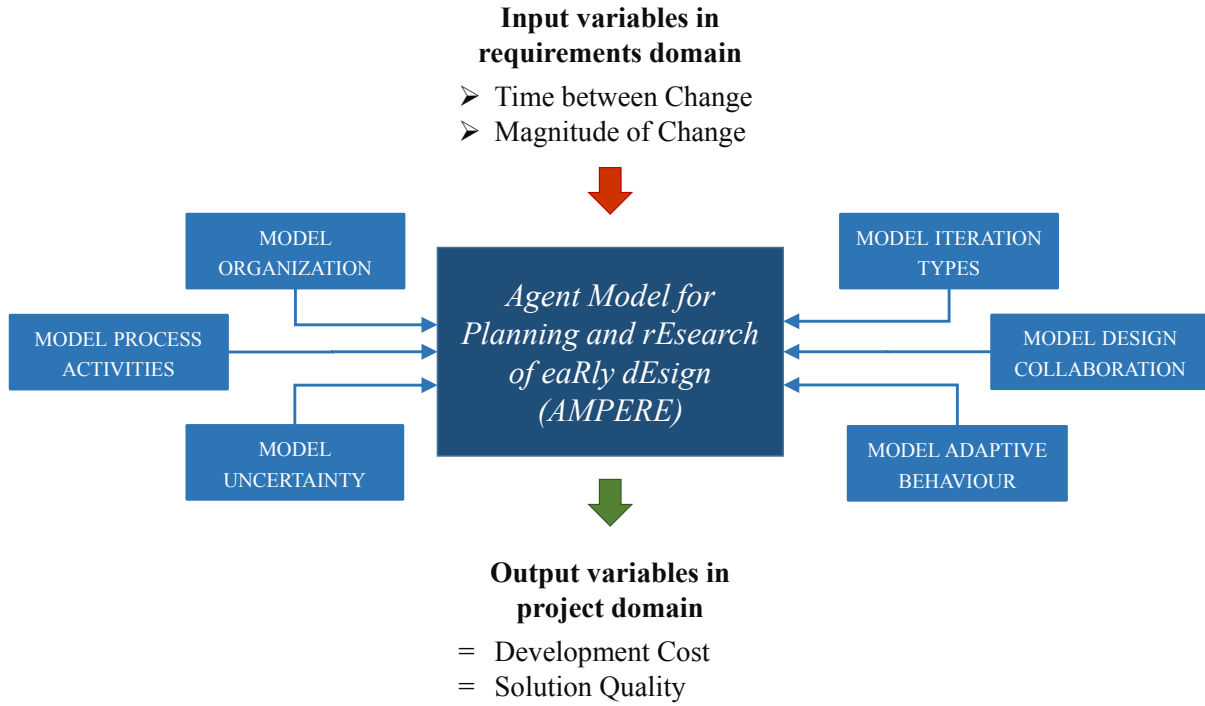


Figure 6.1: Overview of the vision guiding the development of the Agent Model for Planning and rEsearch of eaRly dEsign (AMPERE).

platforms: the difficulty of building agent-based models, i.e., its development cost; and the modelling power offered by the platform, including the ability to scale-up and enhance models in future developments.

The benchmark revealed that dedicated software such as NetLogo and Repast Symphony have been designed to help getting started with agent-based modelling. They allow the construction of models relatively fast for a wide range of applications, providing visual point-and-click tools for agent design, behaviour specification and result visualization (Macal and North, 2011). However, they also generally limit the type of agents that can be built, restrict agent behaviours and communication and possess few libraries supporting the use of mathematical functions within models. Mathematical desktop tools like Mathematica or Matlab provide many capabilities for mathematical modelling and allow some object-oriented programming, but lack specific functionalities for agent-based modelling. Conversely, general purpose object-oriented programming languages like C++, Java or Python offer a full range of agent modelling capabilities, supporting the construc-

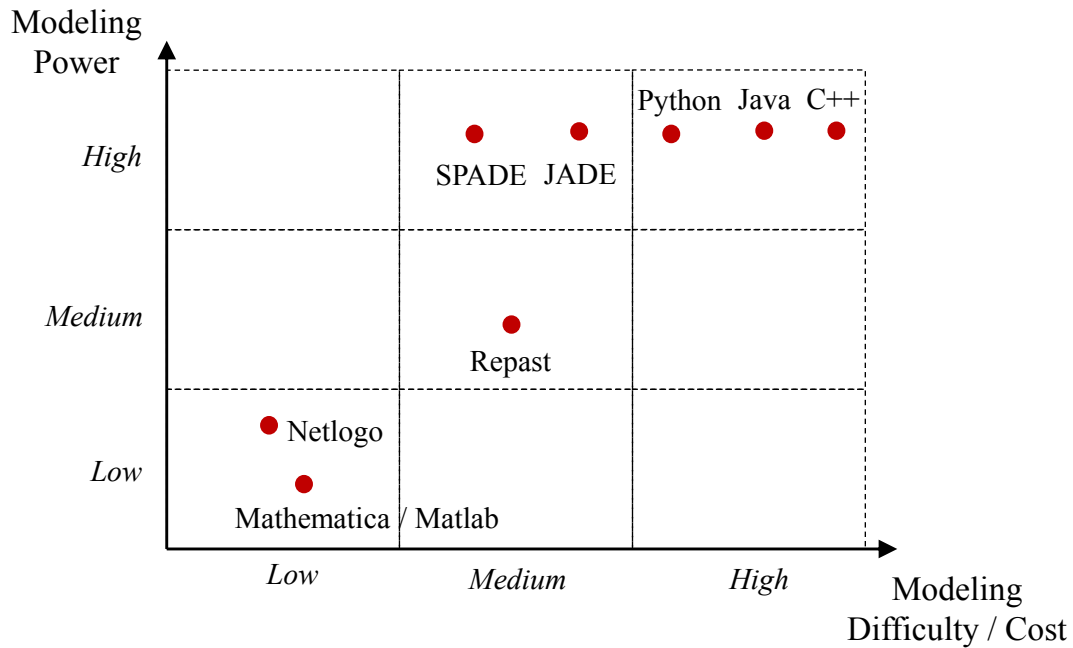


Figure 6.2: Summary of the benchmark made to agent development platforms.

tion of any type of agent and the inclusion of any type of mathematical model. However, it is inherently more difficult to get started and much more expensive to build models from scratch than in dedicated software like NetLogo or Repast.

In order to speed up development using general purpose object-oriented languages, specific platforms have been created for Java – the Java Agent Development Framework, known as JADE (Bellifemine et al., 2001) – and for Python – the Smart Python multi-Agent Development Environment (SPADE), created by Gregori et al. (2006). The aim of both platforms is to provide several services for agent development compliant with FIPA standards, namely (Gregori et al., 2006): an agent management system, which provides a way for agents to register into the platform and be reachable for contact; a directory facilitator, which is a kind of “yellow-pages” where agents can publish services they offer; an agent communication channel, which allows agents to communicate with each other using a content language based on the FIPA-ACL standards.

Figure 6.2 summarizes the results from the benchmark made to the different development tools. Proprietary software for agent-based modelling such as AnyLogic were not evaluated and therefore are not included in Figure 6.2. Aiming for a trade-off modelling

power and development cost, the author found that JADE and SPADE offered the most appropriate development platforms. After analyzing both options, the software platform selected for development of the agent-based model was **SPADE**, since it is Python-based and Python is known for the ability to reduce development costs and allow higher code readability relatively to Java (Lutz, 2009).

### 6.3.3 Architectural design

Having selected the platform for development, the architectural design of the agent model of early design was performed taking advantage of the capabilities already embedded in SPADE. Figure 6.3 presents an overview of the architectural design of the AMPERE based on Unified Modeling Language (UML), which is a language typically used to specify, create and communicate the contents of software (Rumbaugh et al., 1999). A static view is provided in Figure 6.3, representing some of the fundamental classes and the structure of relationships between them. In a static view, each compartment is a class: the top box contains the name, the middle box displays the class attributes and the bottom box communicates the class methods (Rumbaugh et al., 1999).

This static view illustrates that SPADE supports the development of any kind of agents based on several key classes: the SPADE agent class, which has fundamental methods supporting the modeller to setup the agent, add behaviours, register into the platform and to run and shut down the agent, among others; the SPADE ACL Message class, which allows the creation and dispatch of different types of agent communication messages – according to the FIPA-ACL performatives – and the specification of the messages’ content, sender and receiver information; and the SPADE behaviour class, which allows the specification of actions that an agent can perform. Figure 6.3 illustrates that subclasses<sup>11</sup> of the main behaviour class support the creation of different kinds of agent behaviours, namely cyclic and periodic behaviours for repetitive actions, one-shot and time-out behaviours for casual actions, the finite state machine behaviour for capabilities based on internal state transitions, and the event behaviour for actions in response to some event

---

<sup>11</sup>Classes that inherit all attributes and methods from the parent class and can implement variations of the same methods.

that the agent has perceived from the environment at a particular time.

Moreover, Figure 6.3 shows that the model includes specialized agents derived from the general-purpose SPADE “parent” agent. One of the child agents is the **Design Agent**, which was designed as a *practical reasoning agent* (Wooldridge, 2009) and was used to construct further specialized agents based on a common internal structure. Practical reasoning agents use the *Belief-Desire-Intention* (BDI) model of agency proposed by Bratman et al., who defined beliefs, desires and intentions as the three mental components used during rational reasoning directed towards performing actions (Bratman et al., 1998). Beliefs, desires and intentions were perceived as abstract concepts providing a familiar and non-technical way of understanding, modelling and communicating the behaviour of agents and were subsequently used to create software agents with BDI-based architectures (Rao and Georgeff, 1991, 1995).

A practical reasoning agent has internal data structures representing and storing the status of beliefs, desires and intentions and deliberates what action to do using a cyclic algorithm (Wooldridge, 2009) where the agent:

1. Observes the environment and updates his beliefs, which represent the information considered to be true;
2. Selects its desires based on an option generation process;
3. Filters them to select the most appropriate option to commit to;
4. Finds a plan among possible candidates to implement the option selected;
5. And executes the plan.

The Design Agent class shown in Figure 6.3 was architected with attributes representing and storing the agent’s beliefs, desires and intentions. The agent incorporates a “Reasoning” class which implements a practical reasoning algorithm – similar to one described above – which is supported on the class methods for generating options, filtering, planning, executing and revising beliefs. Reasoning is cyclic and defines the agent’s main behaviour. Furthermore, the Design Agent does the planning step through searching into



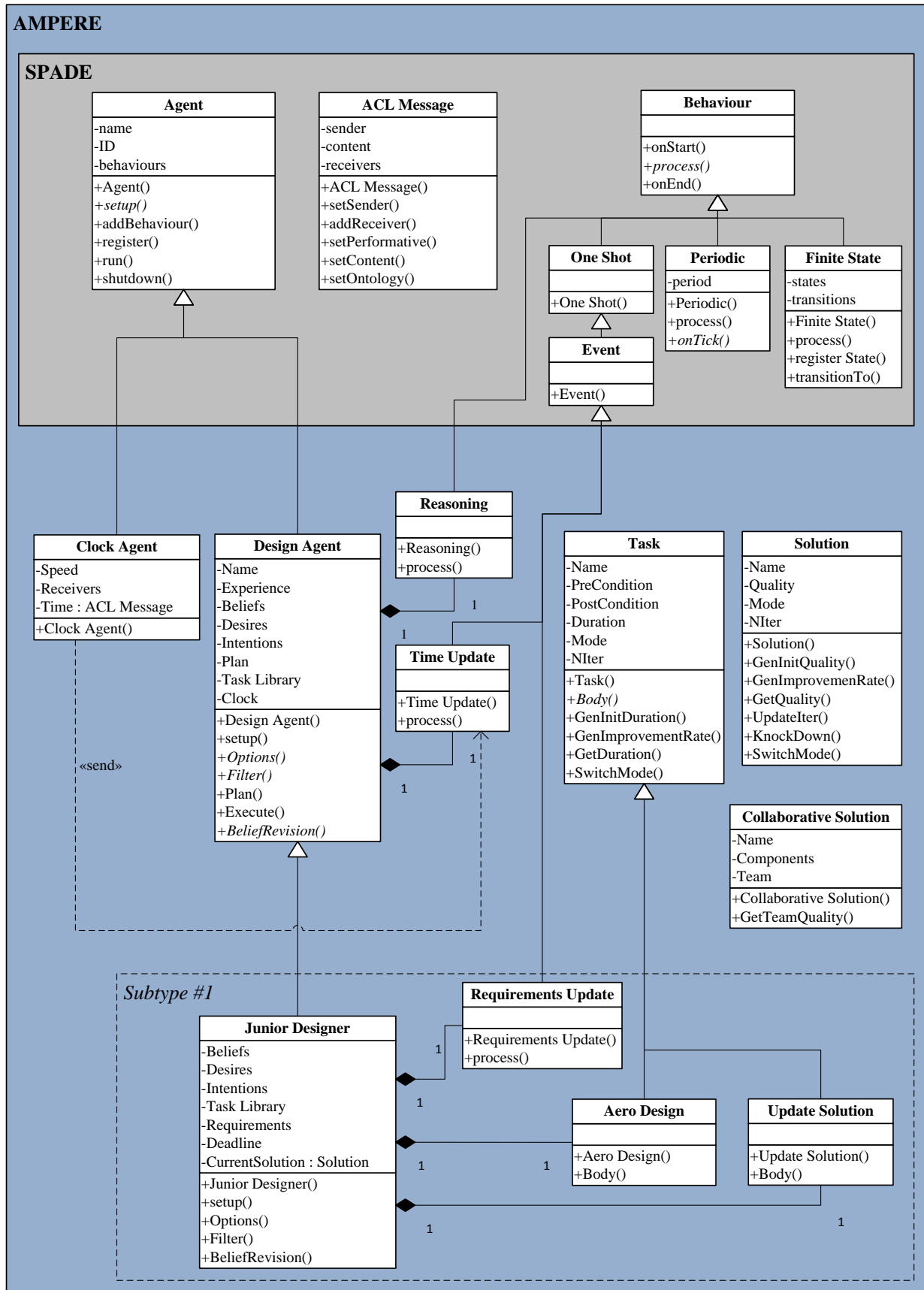


Figure 6.3: Overview of the architectural design of the AMPERE. Static view based on UML containing some of the main classes and the relationships between them. Legend:  $\triangle$  - relationship of inheritance;  $\blacklozenge$  - relationship of composition

an internal library for a task object which has an execution precondition that holds true and a postcondition which matches the goal the agent desires to accomplish. This planning process is similar to the one proposed by Georgeff and Lansky and implemented into the Procedural Reasoning System<sup>12</sup> (Georgeff and Lansky, 1987; Georgeff et al., 1999).

The Task class shown in Figure 6.3 has thus been designed to support the planning process and encode something that a Design Agent is able to do on the environment. Instances of this class or its subclasses have a *precondition* and a *postcondition* that are used by the agent during the Reasoning behaviour. Furthermore, when a Task object is selected from the agent's library as a result of the planning process, the task's *body* method is executed and the agent remains busy – unable to perform any other action – during the time assigned to the *duration* attribute (Figure 6.3). The minimum and maximum duration of a Task object are defined at its construction but the Task class contains methods supporting a dynamic specification of the actual duration according to the status of other attributes, such as the number of times it has been repeated.

Another key class for model construction in AMPERE is the Solution class represented also in Figure 6.3, which represents and tracks the status of the design solution. Aside from other types of tasks, a Design Agent may contain in its library task objects which correspond to the design activities it needs to perform to generate a design solution and improve it over time, such as doing aerodynamic design or mechanical design activities of a system, sub-system or component. The execution of design tasks thus enables a Design Agent to operate and *change* the status of the design solution object it has generated and continuously improve the solution's *quality*. The solution's quality is a key attribute of objects belonging to this class which is used in AMPERE to *parametrize* and represent the level of optimization that the solution has achieved relative to its requirements, through design efforts that have been made on it.

The previous classes were designed to be the main building blocks for developing further specialized agents and integrating them into an agent-model for planning and research of early design. A more detailed description of the internal elements of these classes is

---

<sup>12</sup>A BDI-based framework for constructing real-time reasoning agents that can perform complex tasks in dynamic environments. Found applications in air-traffic and space shuttle systems control.

provided in Appendix C.

The bottom section of Figure 6.3 illustrates also the agent development process with the creation of a simple Junior Designer Agent. It can be noticed from this example that the agent is a subtype of Design Agent having customized versions of the methods for generating options, filtering and belief revision, according to the agent's particular beliefs, desires, intentions and library data structures. Two specialized types of Task were constructed and assigned to the simple Junior Designer's library, encoding in its body the ability to generate an aerodynamic component design and update the solution status into the environment, so that other agents may become aware of it. The solution status is continuously tracked through the agent's *current solution* attribute, which stores an instance generated from the Solution class. In addition, the Junior Designer becomes also aware of the occurrence in the environment of an update of requirements, through the incorporation in the agent's structure of a specific subtype of event behaviour.

The process of creating additional and more sophisticated agents forming an AMPERE model can thus be realized following similar steps. There are, in summary, three major steps: creating a subtype of Design Agent and implementing its own version of option generation, filtering and belief revision capable of processing the agent's particular beliefs, desires, intentions and library data; creating and composing into the agent specific Tasks that it can perform on the environment; and incorporating specific event behaviours that the agent is able to notice and react to. A final section remark is that the architectural design described above has been made with the intent of allowing the creation of many different types of design agents: the classes defined are key general building blocks but are flexible so that they can be subsequently modified and adapted to future developments of the AMPERE.

#### **6.3.4 Agent definition**

Within the scope of this thesis' research questions, the intent for agent-based modelling and for model development has been presented in Figure 6.1: support early design planning and investigate the effects of change in project performance. The agents defined and

described in this section were thus specifically developed to represent a model of early design and capture key dynamics of this stage.

Model development was inspired by the empirical observations discussed in Chapter 4 and illustrated in Figure 4.3 about the early design dynamics of gas turbine development projects. This thesis has previously reported that early project states are governed by frequent interactions between client and supplier organizations. Due to the need to represent such interactions, the author incorporated in the agent model four main subtypes of the Design Agent: a *Customer Agent*, representing the client organization; and a project *Lead Agent*, *Senior Designer Agent* and *Junior Designer Agent*, representing three different types of design agents belonging to the supplier organization. Figure 6.4 illustrates this agent definition.

The Customer Agent incorporates an entity with a privileged view over a particular portion of the environment: the market. This agent is capable of accessing the market to observe how user needs evolve over time and react to events of *change*. Replicating the behaviour illustrated in Figure 4.3, the Customer Agent has an understanding about the requirements that the solution needs to fulfill and is motivated to send regular requests for a design proposal to supplier organizations (Figure 6.4). In addition, the agent is also compelled to trigger an update of requirements when it has observed that the market needs have changed. Conversely, market information is inaccessible to agents belonging to the supplier project organization, but these agents are motivated to respond to requests and updates arriving from the Customer Agent.

Similarly to the type of organizational units found in industrial organizations like Rolls-Royce, the Lead, Senior Designer and Junior Designer agents are the basic building blocks of a *design team* (Figure 6.4). Agents belonging to a design team work together to achieve its goals but there is a hierarchical relationship between the lead and the designers. The Design Lead is modelled with the ability to dispatch *directions* to designer agents belonging to the team. The latter are normally predisposed to accept the directions of the Lead. This agent definition thus also aims to capture the type of social hierarchy observed in

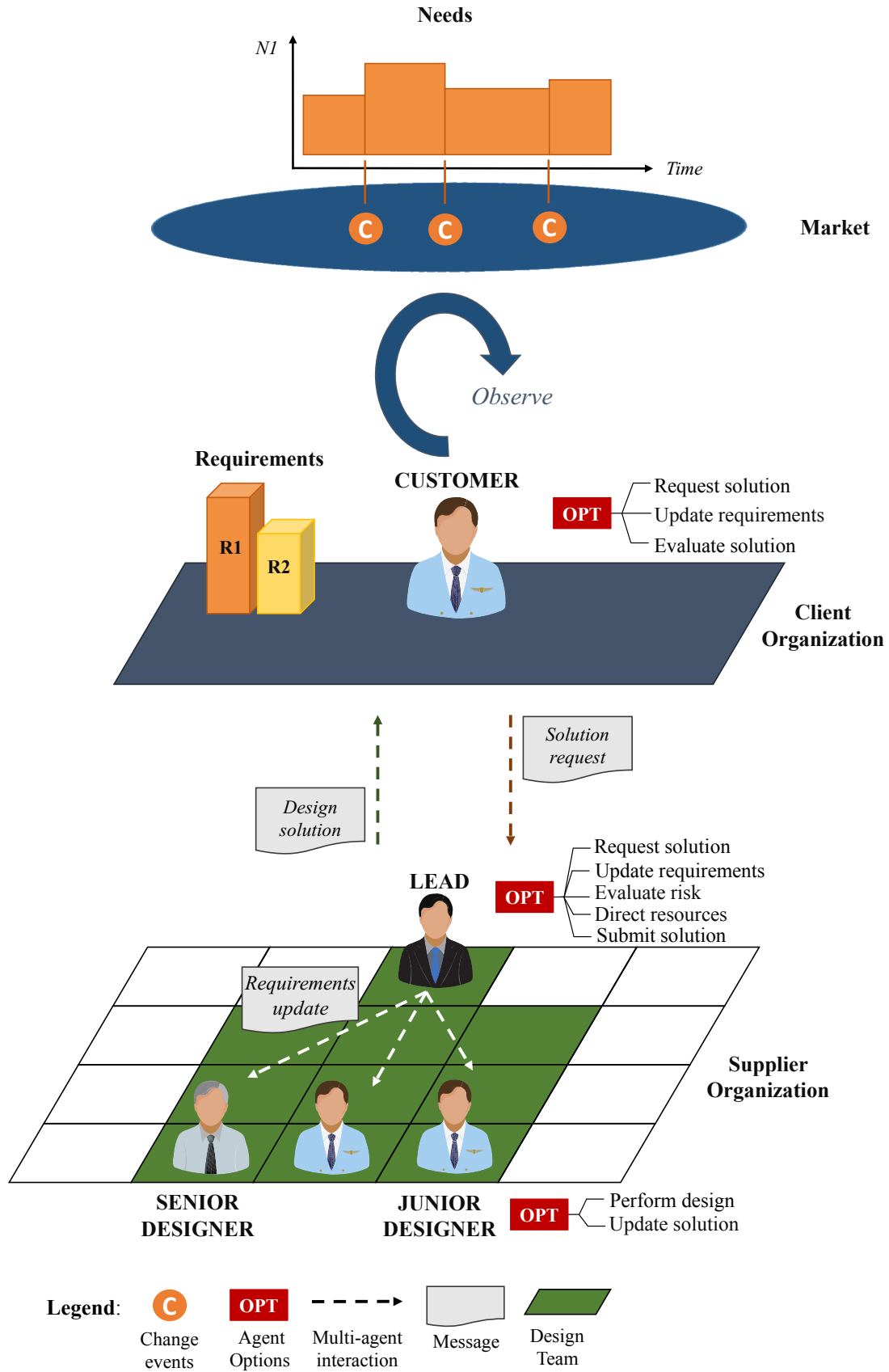


Figure 6.4: Overview of the agent definition in AMPERE, an agent model for planning and research of early design.

practice<sup>13</sup>. Figure 6.4 illustrates that the supplier organization can be composed of one or more design teams working together.

This agent definition also aims to capture distinct levels of design experience which are often used to organize people in design projects, the type of tasks they are allowed to execute and the impact of their actions. The Senior Designer Agent represents engineers who have accumulated a great deal of experience through the completion of many projects and normally participate in early design stages while working across multiple projects at the same time. Due to cost of their resources, senior designers normally participate in collaborative design activities for short periods of time – either regularly or upon request from the project lead – but are able to strongly influence and direct the technical course of the design solution. Conversely, the Junior Designer Agent represents young engineers who have typically completed far less projects. Since they are far less experienced, their resource cost is much lower, compared with senior designers. Empirical observations show that they are normally fully committed to one or two projects at a time and their ability to make the design solution’s performance progress is lower than the one possessed by senior designers.

### **6.3.5 Behaviours and interactions**

Agent behavioral and interaction definitions are also introduced in Figure 6.4. Each agent has the ability to perform a finite number of actions on the environment, including actions that involve interactions with other agents. Actions appear as the outcome of the practical reasoning algorithm described in Section 6.3.3, which consists of updating beliefs, generating options of desires, filtering them, finding a plan that can implement the option selected and executing it.

Figure 6.4 illustrates that the Customer Agent is continuously observing the state of the market and may select between three optional desires: requesting a design solution to the supplier organization; updating requirements to the supplier; and evaluating a design solution received from a supplier. Table 6.1 describes the status of beliefs leading to the

---

<sup>13</sup>The use of the terms “Lead”, “Senior” and “Junior” replicates the terminology used in projects and in design organizations.

Table 6.1: Summary of desire selection according to the status of the agent's beliefs, which supports the specification of agent behaviours in the model.

Agent	Beliefs	Selected Desire(s)
Customer	Time to request solution arrived according to policy	Request solution proposal to supplier
	Market needs changed	Update requirements to supplier
	Received new solution proposal from supplier	Evaluate solution proposal
Lead	Received new requirements	Update requirements to team
	Received new request for solution	Request solution, evaluate risk, direct resources
	Time to replan arrived according to policy	Evaluate risk, direct resources
	Request deadline arrived	Submit solution to customer, direct resources
Junior Designer	Lead requested solution and solution does not meet requirements	Perform design activities
	Lead requested solution and have a solution	Update solution to team
Senior Designer	Lead requested solution and solution does not meet requirements	Support design activities

commitment to one of these desires<sup>14</sup>. For instance, a Customer Agent decides to update its requirements to the supplier organization when it believes that the market needs have changed. Similarly, the Customer chooses to send a request for a solution to the supplier organization when the time defined by its organization's policy has arrived (Table 6.1). The project Lead Agent is able to select between five optional desires: requesting a solution, updating requirements to its design team; evaluating project risks; directing resources, i.e. requesting or allocating designer agents to work on a solution; and submitting a design solution to the Customer (Figure 6.4). For instance, awareness that new customer requirements have arrived triggers the Lead Agent to carry out also a requirements

<sup>14</sup>In practical reasoning, a desire that has been selected among options becomes an intention, i.e., an action that the agent has committed to and is expected to have some desired effect in the environment when it is carried out.

update to the designer agents belonging to its team. Knowledge of the arrival of a new request for a solution causes the agent to implement several actions, namely a solution request, risk evaluation and resource adjustment. Risk evaluation and resource adjustment occur when the agent believes it is time to replan the project's resources according to its internal organizational policy. And solution submission is triggered when the agent believes that the customer's request deadline has arrived.

In addition, Junior Designer Agents can either perform design activities or update the design solution to its team (Figure 6.4). As stated in Section 6.3.3, each designer agent has a collection of tasks in its library that it can perform and that are able to operate on the status of the design solution's quality. The realization and repetition of these tasks – which may relate to real design activities – allows the agent to generate or improve the design solution, similarly to what is empirically observed in the design practice. Table 6.1 shows that the Designer Agent will tend to perform design activities while it is perceived that a solution has been requested and its quality does not yet meet the requirements. Furthermore, recalling the role of senior designers observed in real projects, Senior Designer Agents may decide between supporting the project's design activities or supporting other projects (Table 6.1). The selection of project support leads Senior Designer Agents to engage with other agents belonging to the team – namely with Junior Designer Agents – during the execution of their design activities.

The mapping of the agents' status of beliefs to desires selected for implementation which is summarized in Table 6.1 thus supported the specification of overall behaviour of the different subtypes of Design Agents in the model. As it can be noted in the previous description, some of the actions that can be executed include multi-agent interactions. Multi-agents interactions rely in the exchange of communication messages instances – derived from the SPADE message class illustrated in Figure 6.3 – which allow agents to find each other and pass information through the environment. Table 6.2 summarizes the type of messages exchanged between agents in the model using the FIPA-ACL communication language and describes its content. Table 6.2 describes, for instance, how the communication messages are used by the Customer Agent to request a design solution to



Table 6.2: Summary of communication messages exchanged between agents in the model.

Sender	Receiver	Performative	Content
Customer	Lead	Call-for-Proposal	Time of deadline, solution proposal quality expectation
		Inform	Requirements update
Lead	Customer	Propose	Design solution
	Design Team	Request	Solution, time of deadline, solution quality expectation
		Inform	Requirements update
		Request	Resources needed for project activities
Junior Designer	Design Team	Inform	Design solution update
Senior Designer	Junior Designer	Inform	Start and end of design support

the supplier organization and simultaneously specify two key variables: the deadline for solution submission and the quality that is expected.

The previous definition of agents, behaviours and interactions are the fundamentals of the agent-based model of early design developed by the author with the purpose of investigating the effects of requirements change in project performance. As previously stated, the model is inspired in empirical research about the preliminary gas turbine design and aims to capture key dynamics of early design. The subsequent sections present models of the effects of uncertainty, iteration, collaboration and adaptation introduced in AMPERE to capture such dynamics.

### 6.3.6 Effects of uncertainty and iteration

As previously stated, each subtype of Design Agent is built in AMPERE with a library of specialized tasks it can choose to perform. For Junior and Senior Designer Agents, the execution of tasks consisting of actual design activities enables the agents to generate a design solution instance or operate and change its status after it has been created. This allows designer agents to gradually improve their knowledge and thus the quality of the

design solution object.

As shown in Table 6.1, designer agents become aware – within their database of beliefs – of what is the solution quality expected upon the reception of a request for a solution. This knowledge enables them during their practical reasoning process to decide to iterate design tasks until they perceive that the target quality has been achieved. The effects of design iteration on the agent’s design solution instance *quality attribute* is modelled mathematically in AMPERE according to Equation 6.1:

$$Q(n) = Q_s - (Q_s - Q_i)e^{-\alpha n} \quad (6.1)$$

where  $Q$  - is solution instance quality attribute;  $n$  - is the number of accumulated iterations;  $Q_s$  - is the standard quality level the designer agent is able to achieve;  $Q_i$  - is the initial quality level; and  $\alpha$  - is the quality progress rate coefficient. During simulation,  $Q$ ,  $Q_s$ ,  $Q_i$  and  $\alpha$  take real values between  $[0, 1]$  while  $n$  stores integers within the interval  $[0, +\infty]$ . Considering the concept of quality as a representation of the degree of solution optimization, a unitary value thus represents a fully optimized design solution.

Similarly, the effect of iteration on the agent’s design task instance *duration attribute* is modelled in AMPERE according to Equation 6.2:

$$D(n) = D_s + (D_i - D_s)e^{-\beta n} \quad (6.2)$$

where  $D$  - is task instance duration attribute;  $n$  - is the number of accumulated iterations;  $D_s$  - is the standard duration the designer agent is capable;  $D_i$  - is the initial duration of the task; and  $\beta$  - is the duration progress rate coefficient. Durations  $D$ ,  $D_s$  and  $D_i$  are defined as real numbers corresponding to time units,  $n$  as integers between  $[0, +\infty]$  and  $\beta$  takes real values within  $[0, 1]$  during simulation.

Equations 6.1 and 6.2 thus propose to model the effect of design iteration on the solution’s quality attribute and on the task’s duration attribute through exponential learning curves. The *exponential learning curve* has been previously reported across several domains, such as in cognitive psychology, in natural sciences or in manufacturing, as one of the standard

ways to describe the way improvements in the performance of human tasks evolve with practice (Yelle, 1979; Heathcote et al., 2000; Leibowitz et al., 2010). The exponential law of practice essentially describes in mathematical terms that further practice of a task improves performance, but yields increasingly lower performance gains. Empirical studies reported by Hamade et al. suggest that a similar learning behaviour takes place with engineers carrying out design activities (Hamade et al., 2005).

The exponential law parametrized in Equation 6.1 allows the agent model to incorporate that, for the same level of activity fidelity<sup>15</sup>, the first design iterations performed by the agent yield large improvements in the design solution’s quality. However, as the agent accumulates more repetitions of the design tasks, the increase in quality diminishes quickly, until it becomes marginal. Similarly, the law parametrized in Equation 6.2 incorporates in the model that practice accumulation increases efficiency and enables designers to reduce task durations with activity repetitions, for a constant level of fidelity. However, as repetition accumulates, efficiency gains become increasingly more difficult to deliver.

These behaviours reproduce two empirical notions. The first is that the level of solution optimality relative to requirements – represented in the model by the “quality” attribute – grows faster during early stages of design exploration and convergence and as the design process progresses to stages of refinement and repetition, it requires increasingly higher effort from designers to continue optimizing the solution. And the second is that designers learn from experience accumulation and this allows them to perform subsequent repetitions of activities faster than initial executions. The previous behaviours of quality and duration improvement with iteration accumulation have been incorporated in AMPERE through the methods created for the Task and Solution classes (Figure 6.3).

Figures 6.5(a) and (b) represent the effects incorporated through the use of Equations 6.1 and 6.2 and show that the model allows each agent to have particular laws of performance improvement, according to its level of *design experience*. The agent model thus also aims to capture the empirical notion that designers with higher accumulated experience will be

---

<sup>15</sup>Fidelity relates to precision in results and confidence in the knowledge generated from activities about the solution’s behaviour. Increasing levels of fidelity arise from increasing sophistication in methods and tools used by designers during design activities. An example of alternative levels of fidelity is the use of 1D, 2D or 3D calculation methods in gas turbine aerodynamic or mechanical design activities.

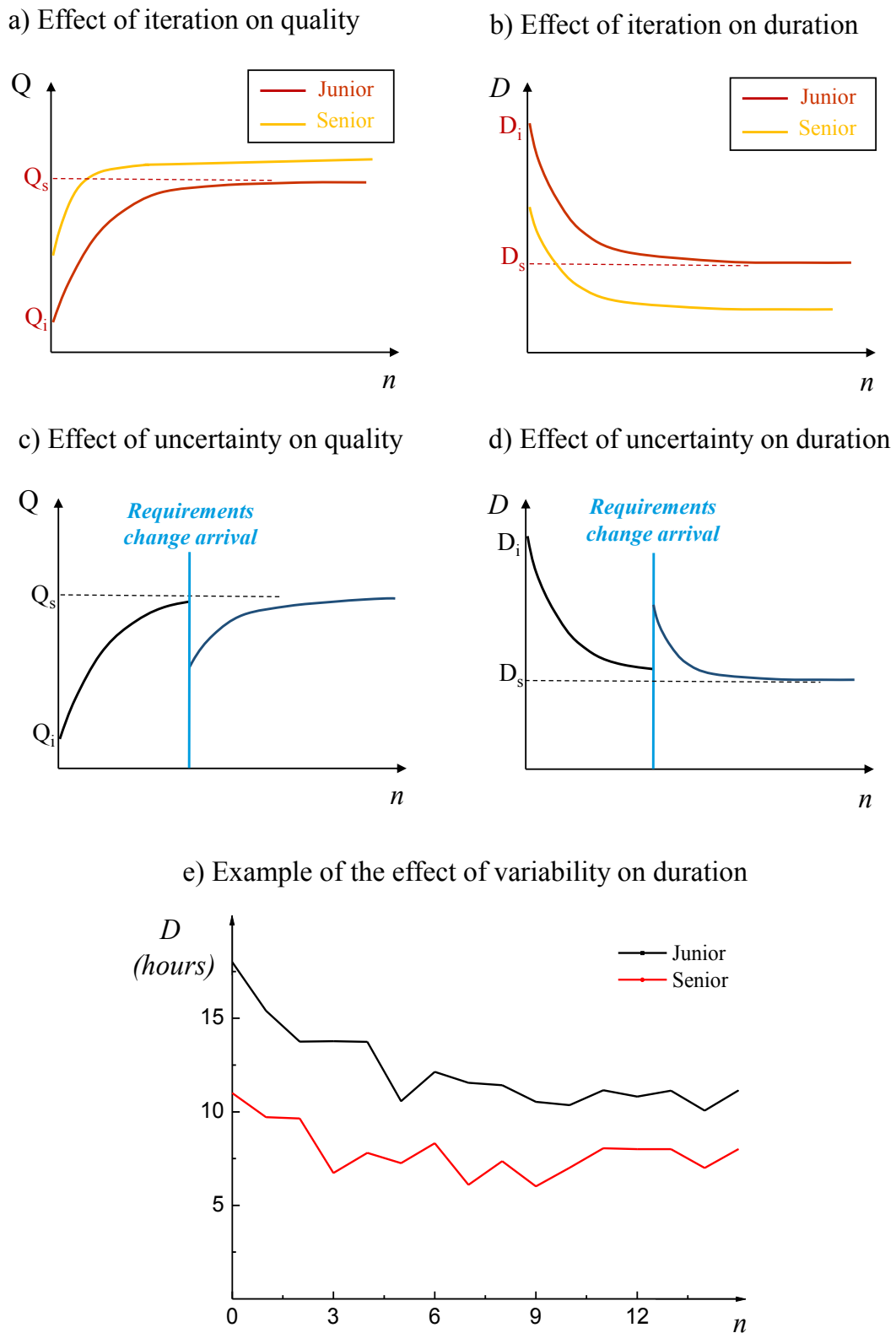


Figure 6.5: Effects of uncertainty and iteration in AMPERE: a) quality and b) duration, according to iteration and agent experience; c) quality and d) duration, according to events of change; e) example of variability effects on duration.

typically capable of generating solutions with higher quality levels or improving it faster than those with lower experience. Likewise, designers with lower experience tend to need more time to accomplish activities than those more experienced. The AMPERE allows the prescription of such behaviours through the definition of different parameters in the exponential laws to Senior Designer and Junior Designer agents.

Uncertainties typical of the design activity are also key factors influencing the ability of agents in the model to arrive to a satisfactory solution. As shown in Figure 6.4, external uncertainties such as events of change in market needs have been included in the AMPERE and drive the Customer Agent to send updates of requirements to the supplier's organization. Figures 6.5(c) and (d) illustrate that, when agents perceive the arrival of a change in requirements, the event causes a deterioration in the level of performance that the agent has achieved until that moment. This deterioration is experienced in terms of a loss of solution quality and design work efficiency, which aims to reproduce the effects resulting from the repositioning of the solution in the design space that occurs when requirements change. This deterioration effect intends to capture the empirical conception that change transports designers to a state of lower knowledge, since the goals have effectively been modified relatively to the solution status. The “amount” of solution quality and activity efficiency loss has been modelled in AMPERE proportionally to the *magnitude* of the change perceived by the agent.

Moreover, in order to account for the effect of day-to-day *variability* in individual performance of agents, Equations 6.1 and 6.2 have been implemented in the agent model with probability density functions associated to the standard value, the initial value and to the progress rate coefficient. For instance, the standard task duration value –  $D_s$  in Equation 6.2 – is sampled during simulation when the agent executes a task  $T$  according to:

$$D_s^T = D_s^{T@construction} \times Tri(C_{low}, C_{high}, C_{mode}) \quad (6.3)$$

where  $D_s^T$  - is the standard duration of the task at simulation time  $t$  when the agent executes the task;  $D_s^{T@construction}$  - is the standard duration specified for the task object at construction; and  $Tri$  - is a triangular probability density function with lower and higher

bounds  $C_{low}$  and  $C_{high}$ , respectively, and a mode given by  $C_{mode}$ . These are constants – for instance  $C_{low}, C_{high}, C_{mode} = 0.8, 1.2, 1.0$  – defining the variability of the standard duration relatively to the value defined at setup time. Figure 6.5(e) represents the evolution in standard task duration with the number of task iterations when variability is introduced. It thus illustrates the effect of Equation 6.3 on the exponential type of learning behaviour modelled through Equations 6.1 and 6.2.

AMPERE uses simple probability density functions, such as the triangular functions, to capture the effects of day-to-day variability during simulation. The model allows the specification of different probability density functions for the standard value, initial value and progress rate coefficients defined in Equations 6.1 and 6.2 in order to account for differences in the consistency of performance due to varying levels of experience. An illustration of the evolution of activity duration for Junior and Senior Designers taking into account variability is also provided in Figure 6.5(e).

### **6.3.7 Effects of collaboration and adaptation**

In addition, two different facets of collaboration between actors often observed during complex design processes have been also incorporated in AMPERE. One fundamental dimension of collaboration is accounting for the traditional breakdown of large pieces of design work into separate and smaller parts, which are delivered by different actors. To accommodate the work breakdown principles, AMPERE allows the definition – through the agent’s task library – of specific design responsibilities to individual Senior or Junior Design Agents, such as component or discipline design responsibilities. For instance, the agent model allows one Junior Designer Agent to be responsible for the aerodynamic design of one specific component and require this agent to pass information to another Junior Designer Agent part of his team, which is responsible for the mechanical design of the same component. Both work until they are satisfied with their individual solutions, reacting to changes that are communicated during that process.

Since this facet of collaboration requires partial solutions to be integrated into a more global solution, the architectural design of AMPERE includes an additional solution class,

which has been coined as the Collaborative Solution class (Figure 6.3). An object of this class tracks and stores the results from a collaborative design process where a set of agents belonging to the same or different design teams are working separately on different components or disciplinary aspects of a more global design solution. The quality of the collaborative design solution instance is defined within this class according to Equation 6.4:

$$CSQ = \sum_{j=1}^n w_j Q_j \quad (6.4)$$

where  $CSQ$  - is the collaborative solution instance quality attribute;  $w_j$  - is the weight of each part or aspect for the global design solution quality and  $Q_j$  - is the design solution instance quality attribute of each part or aspect, being  $w_j$  subject to the constraint that:

$$\sum_{j=1}^n w_j = 1 \quad (6.5)$$

During simulation,  $CSQ$ ,  $Q_j$  and  $w_j$  take real values between  $[0, 1]$ . Equations 6.4 and 6.5 capture mathematically the effects of collaboration based in work decomposition. The Lead Agent is typically an holder of a Collaborative Solution instance since it must monitor the evolution of the work made by Senior and Junior Designers belonging to his team during the design process of their individual solutions.

Moreover, being agent-based model where concurrency is easy to implement, AMPERE allows also that different Designer Agents are assigned to develop alternative design solutions for the *same* component or disciplinary aspect of the solution. This intends to capture the dynamics of more types of iteration, such as exploration and convergence where designers work concurrently on several candidates for the solution (or partial solution). The Collaborative Solution class shown in Figure 6.3 has been developed with methods that allow agents to screen among concurrent solutions developed with the same goal and down select the one that has achieved the most promising result, i.e., the *maximum* quality level.

The second dimension of collaboration that has been incorporated in the model is designer-to-designer support. This facet of collaboration occurs often in practice when more ex-

perienced engineers dedicate part of their project time supporting the activities of less experienced ones, thus contributing to solving problems and improving the design solution's quality at a faster pace. Table 6.2 describes that multi-agent interactions consisting of support provision to Junior Designers is one of the options made available to Senior Designer Agents. The effect of Senior engagement and support provision in the performance of the output arising from the work made by Junior Designers has been modelled through a *temporary* increase in their quality progress rate coefficient, the parameter  $\alpha$  in Equation 6.1. This increase persists during the time allocated between the agents for collaborative work.

Several facets of adaptive decision-making behaviour encountered during complex design have also been incorporated in the agent model. The practical reasoning behaviour used by agents to plan their actions according to the state of the environment is, in essence, a way of capturing in-situ decision-making. The previous dimension of collaborative behaviour also includes adaptation, since Senior Designer Agents have been made capable of deciding to support less experienced designers when they observe that a request has been sent but the design solution does not yet met the expectations.

Another key example of adaptive behaviour included in the model arises from the actions which have been made available to the Lead Agent. This agent is capable of evaluating risks and deciding either to direct resources available in the design team to work on project activities or not. Table 6.2 shows these are performed regularly and thus they constitute effectively a behaviour of *adaptive planning* to the project's needs. The task of risk evaluation made by the Lead Agent includes in-situ estimation of the probability and impact of not meeting the Customer's expectations. Probability and impact are estimated real numbers between  $[0, 1]$  and risk is computed by the agent using the standard approach of  $Risk = Probability \times Impact$ . The probability of not meeting the Customer's expectations has been modelled in AMPERE according to Equation 6.6 as a function of the time available until the deadline provided by the Customer and the gap in quality



relative to the Customer's expected level:

$$Probability(T_{gap}, Q_{gap}) = 1 - e^{(c_2 T_{gap} / T_{deadline} - c_1) \times Q_{gap}} \quad (6.6)$$

where  $T_{deadline}$  - is the time corresponding to the deadline defined by the Customer agent;  $T_{gap}$  - is the difference between  $T_{deadline}$  and the present time perceived by the Lead agent;  $Q_{gap}$  - is the difference between the current level of solution quality achieved by the team and the level expected by the Customer agent; and  $c_1$  and  $c_2$  - are real constants adjusting the rate of evolution of the probability function. During simulation,  $T_{deadline}$  and  $T_{gap}$  are real numbers corresponding to time units while  $Q_{gap}$  takes a real value between  $[0, 1]$ .

The impact of failing to deliver has been modelled in AMPERE according to Equation 6.6 simply as a linear function of the gap in quality relative to the Customer's expectations:

$$Impact(Q_{gap}) = \begin{cases} m \times Q_{gap} & \text{if } Q_{gap} \leq Tol \\ 1 & \text{if } Q_{gap} > Tol \end{cases} \quad (6.7)$$

where  $m$  - is a real constant adjusting the rate of evolution of the impact function; and  $Tol$  is a real number between  $[0, 1]$  defining a tolerance level above which the agent is no longer sensitive to the quality gap and the impact is maximum.

As stated above, the Lead agent performs resources adjustments subsequently to an internal evaluation of the risk of failing to deliver on time and with the expected level of quality. Resources are adjusted by the Lead agent in a linear manner, based on the level of risk perceived at that time. Higher levels of perceived risk impel the Lead Agent to send a request for design work to more resources available in his team. Conversely, a decrease in risk levels occurring as a result of design efforts made by the team drives the Lead to cancel previous orders and free resources in the team.

## 6.4 Exploratory simulations

This section presents and evaluates results arising from simulation of an initial AMPERE model created by the author which conceptualizes a simple scenario: a single customer which engages with a single supplier organization comprising only one design team responsible for delivering a solution to the customer. Due to this simplicity, these initial simulations can be considered as *exploratory*. Results from these simulations are thus *illustrative* and provide essentially an *initial* evaluation of the potential of the agent model to capture key dynamics of early design stages, investigate the effects of expectable levels of change in project performance and support planning. The section begins by describing the setup of this simple model used for exploratory simulations.

### 6.4.1 Simulation setup

Based on the empirical research of early stages of gas turbine design previously described (Section 4.4.1), initial simulations were supported in a simple agent model consisting of a single Customer agent which regularly sends a request for a solution proposal to a single supplier. Within this organization, a single design team composed of a Lead agent and several Senior and Junior Designer agents is in charge of generating a design solution proposal that meets the Customer's expectations and responds to the Customer before the specified deadline. In addition, changes occurring in the market environment that the Customer is continuously observing generate updates of requirements sent to the design team. This team then needs to be capable of adapting its resources and activities to tackle the arrival of changes.

The setup of this simple AMPERE model for simulation essentially included the definition of specific internal parameters adjusting the behaviour of the environment and the general behaviour of the design agents – previously described in Tables 6.1 and 6.2 – to further represent the dynamics of early design stages. Starting with the external environment, the initial model has been setup with the generation of change events in the market observed by the Customer with a period uniformly distributed between 10 and 20 working days – i.e.,  $TbC = Unif(10, 20)$  – and a magnitude of change following a similar distribution

between 1 and 10%, i.e.  $\text{MoC} = \text{Unif}(1, 10)$ .

The additional setup for agents is introduced in Table 6.3. It shows that the Customer agent was setup with an internal policy defining the desire to request a new solution proposal every 4 working weeks. When a request is sent, the Customer expects a response from the supplier one working day before the time to request a new proposal has arrived. Table 6.3 shows also that the Customer was setup with the aim to get a certain number of solution proposals. After this pre-determined number is achieved, the Customer agent has achieved its goal and the simulation stops. This setup aims to capture the dynamics reported in Section 4.4.1 for early design stages, where the Customer drives the suppliers' design process through regular requests for proposals until it makes a decision about which solution is most promising.

Tables 6.1 and 6.2 defined that Junior Designer agents were essentially capable of performing design activities to generate a solution and update its status to the design team, while Senior Designer agents were more experienced individuals engaged in multiple projects that could be requested to support design activities during specific time intervals. To further capture the dynamics of design processes, the agent model was setup for simulation with a design team composed of two groups of discipline designers, one responsible for aerodynamic design and the other for mechanical design of the system requested.

Table 6.3 shows that aerodynamic designers have the ability to assess the requirements received, generate an aerodynamic concept, create a model to compute the efficiency of the concept and evaluate the solution that they have created. Mechanical designers are also capable of assessing new requirements and, in addition, assess updates of the aerodynamic solution generated by fellow designers in order to compute the loads acting on the system. After loads have been computed, mechanical designers generate a structural model and compute the stress, weight and cost of the design solution. Each of the design tasks performed by discipline designers was setup with specific duration parameters –  $D_s$  and  $D_i$  in Equation 6.2 – and an associated task duration variability –  $C_{low}$ ,  $C_{high}$ ,  $C_{mode}$  in Equation 6.3 – which are summarized in Table 6.3. Activity durations reported in Table

Table 6.3: Summary of the specific setup of the agent model for exploratory simulations.

Agent	Behaviour	Characteristics
Customer	Request solution to supplier	<ul style="list-style-type: none"> <li>• Policy: every 4 working weeks</li> <li>• Deadline: 1 day before next request</li> <li>• Expected quality level: 1.0</li> <li>• Maximum number: 3</li> </ul>
Lead	Evaluate risk, direct resources	<ul style="list-style-type: none"> <li>• Policy: every working week</li> </ul>
Junior Designer, Aerodynamics	Perform design activities (1-5): <ol style="list-style-type: none"> <li>1. Assess requirements</li> <li>2. Generate aero concept</li> <li>3. Generate aerodynamic model</li> <li>4. Compute efficiency</li> <li>5. Evaluate solution</li> </ol>	Duration setup parameters: <ol style="list-style-type: none"> <li>1. <math>D_s = 1h</math>; <math>D_i = 4h</math></li> <li>2. <math>D_s = 4h</math>; <math>D_i = 12h</math></li> <li>3. <math>D_s = 2h</math>; <math>D_i = 6h</math></li> <li>4. <math>D_s = 3h</math>; <math>D_i = 8h</math></li> <li>5. <math>D_s = 1h</math>; <math>D_i = 4h</math></li> </ol> Duration variability parameters: <ul style="list-style-type: none"> <li>• <math>C_{low} = 0.9</math>; <math>C_{high} = 1.4</math>; <math>C_{mode} = 1.0</math></li> </ul> Quality parameters: <ul style="list-style-type: none"> <li>• <math>Q_i = Tri(0.1; 0.4; 0.2)</math>; <math>Q_s = 1.0</math></li> </ul> Progress rate parameters: <ul style="list-style-type: none"> <li>• <math>\alpha = Tri(0.1, 0.3, 0.2)</math>;</li> <li>• <math>\beta = Tri(0.2; 0.6; 0.3)</math>;</li> </ul>
Junior Designer, Mechanical	Perform design activities (1-6): <ol style="list-style-type: none"> <li>1. Assess requirements</li> <li>2. Assess aero concept</li> <li>3. Compute loads</li> <li>4. Generate structural model</li> <li>5. Compute stress, weight, cost</li> <li>6. Evaluate solution</li> </ol>	Duration setup parameters: <ol style="list-style-type: none"> <li>1. <math>D_s = 1h</math>; <math>D_i = 4h</math></li> <li>2. <math>D_s = 2h</math>; <math>D_i = 6h</math></li> <li>3. <math>D_s = 3h</math>; <math>D_i = 9h</math></li> <li>4. <math>D_s = 2h</math>; <math>D_i = 7h</math></li> <li>5. <math>D_s = 2h</math>; <math>D_i = 5h</math></li> <li>6. <math>D_s = 1h</math>; <math>D_i = 4h</math></li> </ol> Duration variability parameters: <ul style="list-style-type: none"> <li>• <math>C_{low} = 0.9</math>; <math>C_{high} = 1.4</math>; <math>C_{mode} = 1.0</math></li> </ul> Quality parameters: <ul style="list-style-type: none"> <li>• <math>Q_i = Tri(0.1; 0.4; 0.2)</math>; <math>Q_s = 1.0</math></li> </ul> Progress rate parameters: <ul style="list-style-type: none"> <li>• <math>\alpha = Tri(0.1, 0.3, 0.2)</math></li> <li>• <math>\beta = Tri(0.2; 0.6; 0.3)</math></li> </ul>
Senior Designer, Aerodynamics / Mechanical	Support design activities	Collaboration parameters: <ul style="list-style-type: none"> <li>• <math>\alpha = Tri(0.2, 0.4, 0.3)</math>;</li> <li>• <math>\beta = Tri(0.4, 0.7, 0.5)</math>;</li> </ul>

6.3 refer to hours of working time<sup>16</sup>. In addition, in order to account for the appearance of mistakes during task execution which lead to task rework, each design task was setup with a probability of rework equal to 10%.

Triangular probability density functions were assigned to the solution quality and duration progress rate coefficients – respectively  $\alpha$  and  $\beta$  in Equations 6.1 and 6.2 – according to the agent’s level of experience, as shown in Table 6.3. The effects of collaboration provided by a Senior Designer agents can be observed in Table 6.3 through the setup of higher progress rate coefficients, compared to the values achieved by Juniors during activities performed in an individual mode. The initial solution quality,  $Q_i$  in Equation 6.1, achieved by the designer agents after the first design iteration has been made is also sampled from a triangular distribution.

Tables 6.1 and 6.2 described also that the Lead agent has the ability to request solutions to the designers in his team, update requirements when they have changed, respond to the Customer and continuously evaluate the technical risk of the project and adjust the level of resources required to meet the expected solution quality before the deadline arrives. Table 6.3 shows that the Lead agent was setup for simulation with an internal policy defining the desire to perform risk evaluation and resource adjustments every working week.

In this simple model, the Lead agent has an available design team composed of five designer agents: two aerodynamic Designers, one Junior and one Senior; and three mechanical Designers, where two are Junior level and one is Senior level. The weight,  $w_j$  in Equation 6.4, assigned by the Lead agent to each disciplinary design solution is equal to 0.5 during the generation of a collaborative solution. As described in Tables 6.1 and 6.2, when designers are requested to generate a solution through design work, they actively seek to fulfill the Lead agent’s request. A simplification present in this initial model for exploratory simulations is thus that agents are not delayed due to commitments arising from other design projects and act immediately upon request. When work is performed, agents contribute to the project’s development expenses with an hourly cost defined ac-

---

<sup>16</sup>During simulation, the model considers that agents work on average 22 days/month and 8 hours/day.

according to the agent's level of experience. In this setup, Junior and Senior Designer agents were assigned an hourly rate of 25 and 75 cost units/hour, respectively.

Simulation begins with agents registering into the SPADE platform, which allows them to be reachable for contact and exchange messages through the platform's communication channel. Simulation is driven by an initial request for a solution sent by the Customer agent to the supplier, which occurs at the simulation's initial time step. A time-stepped approach has been implemented in AMPERE and the simulation's internal clock advances at constant time intervals. This clock was included in the architectural design of AMPERE as a SPADE agent which continuously sends out the current time step to all Design Agents (Figure 6.3). This ensures time-stepping synchronization during simulation.

This initial model used in exploratory simulations requires approximately 2500 lines of code to run, including the classes and methods represented in Figure 6.3 and described in Section 6.3.3 as the heart of AMPERE. It takes 25–35 seconds to run a complete simulation of this initial agent model on an Intel Core i7-3517 CPU@1.9 GHz using a 64-bit operating system.

### **6.4.2 Process visualization**

Section 5.1 referred that process visualization is a key purpose of product development process modelling since it provides support for group discussions within design organizations. Because of that, the author used exploratory simulations to investigate ways to picture the design process resulting from agent-based simulations. The generation of a design process chart has been included as one of the standard post-processing features for AMPERE simulations and provides a visualization tool to analyze the process activities performed by agents during simulation runs.

Figure 6.6 presents an example of the complete design process chart resulting from a particular simulation run with the setup described in Section 6.4.1. An overview of this chart shows that the overall design process duration was approximately 12 weeks, following the proposal request policy incorporated in the Customer agent. The policy determined in

this case three major design iteration cycles with the supplier. A high-level analysis of Figure 6.6 reveals also that several changes occurred in the market environment during the simulation time, which are visible through the various requirements updates sent from the Customer agent to the supplier.

These changes perceived by the Customer are also highlighted in Figure 6.7, which depicts a more detailed view of the same design process chart and supports further analysis over the agents' actions. For instance, patterns of risk evaluation and resource adjustments are identified in Figure 6.7, which result from the Lead agent's internal policy of regularly replanning according to the project's needs. Directions arising from the Lead agent trigger Designer agents to start their design activities. Figure 6.7 shows that the Lead agent often requests additional resources to support the project and highlights that Junior Designers engage in concurrent design iterations while Senior Designers provide design support to Juniors upon the Lead's request.

Activity patterns appear also as a result of the Designers' need to iterate the design solution to further improve its quality. Figure 6.7 points out the repeating patterns arising from aerodynamic design iterations. These consist in the generation of an aero concept and model for performance calculation, followed by computation of solution efficiency, evaluation and solution update to other teams.

In addition, occasional interruptions of the natural sequence of activities composing an iteration cycle appear as a result of reactive agent behaviours, which consist of reactions to changes arriving from the surroundings. Reactive behaviours performed by Designer agents are also visible with more detail in Figure 6.7. These include, for instance, in-situ stoppage of other ongoing design activities of an iteration to perform a re-assessment of requirements upon the perception of a requirements change event. Or a reaction to the arrival of solution updates from other teams, which drives for instance mechanical Designers to decide for the re-assessment of the concept generated by the aerodynamics design team when an update is communicated by aerodynamic Designers (Figure 6.7).

Furthermore, looking back to the comparison between alternative product development modelling approaches, the visualizations depicted in Figures 6.6 and 6.7 reveal some of

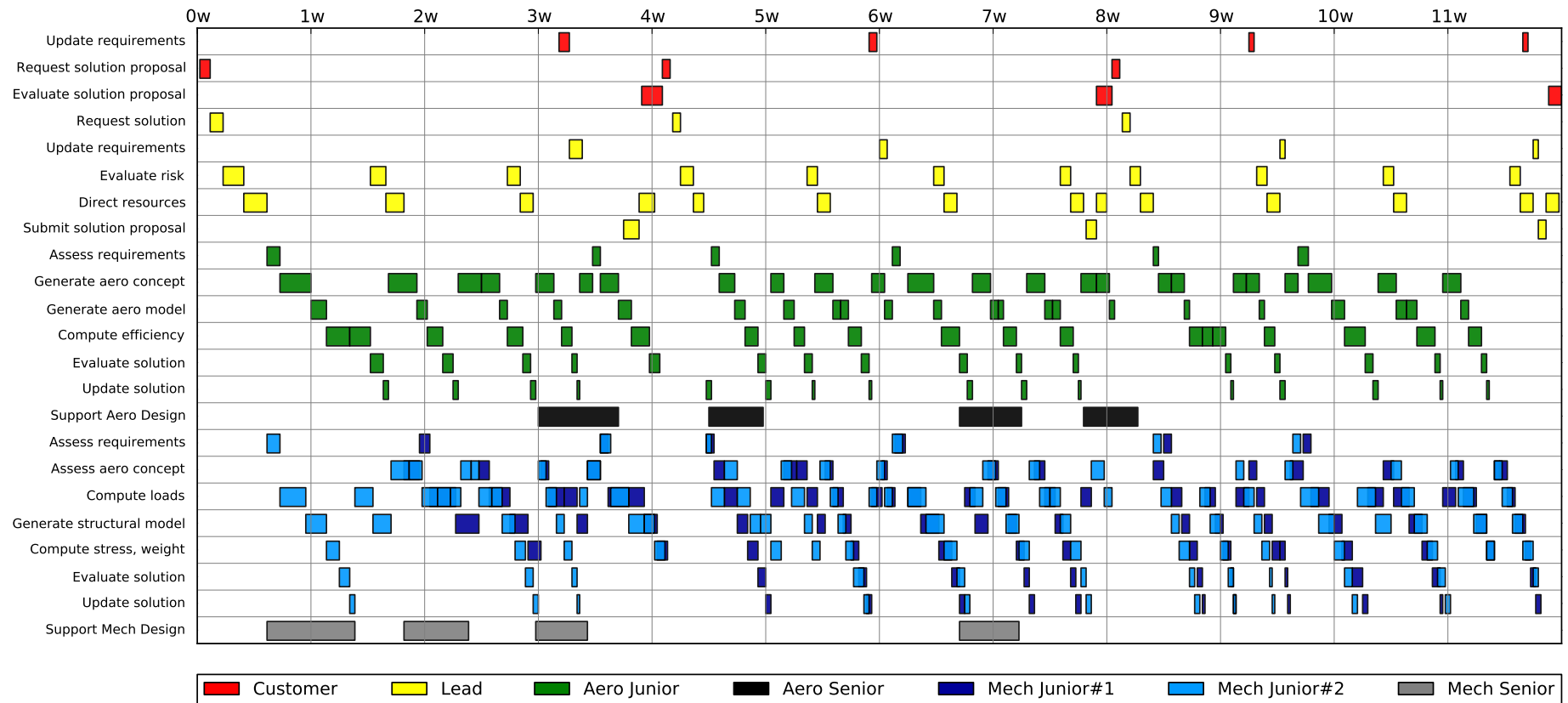


Figure 6.6: Complete design process chart arising from a single simulation run. Supports visualization of the activities executed by agents during simulation.



the fundamental strengths of agent-based simulation. It becomes apparent from these visualizations how the overall design process *emerges* from concurrent streams of design activities performed by different agent teams, how the process is shaped by frequent social interactions between agents and how the final activity network – largely pre-determined in activity-based models – is also influenced by adaptive behaviours which have been incorporated in the agents.

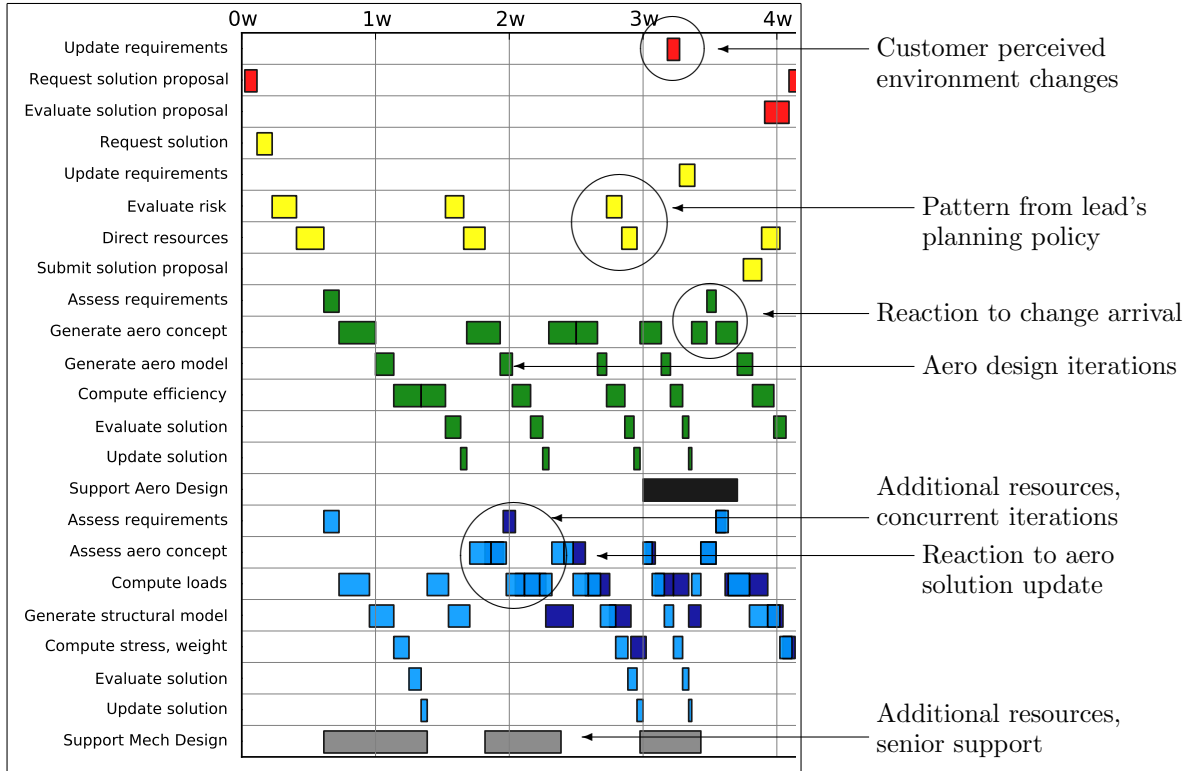


Figure 6.7: Detailed view with annotations of the design process chart arising from a single simulation run.

### 6.4.3 Time evolution of internal variables

In addition to process visualization, post-processing of AMPERE simulations includes also the visualization of the evolution of internal agent variables which have an important influence in the outcome of the simulation. Standard post-processing examples explored during these initial simulations are shown throughout Figures 6.8, 6.9 and 6.10.

The evolution of the solution quality generated by designer agents and perceived by the Lead during the course of one particular simulation run is presented in Figure 6.8. This

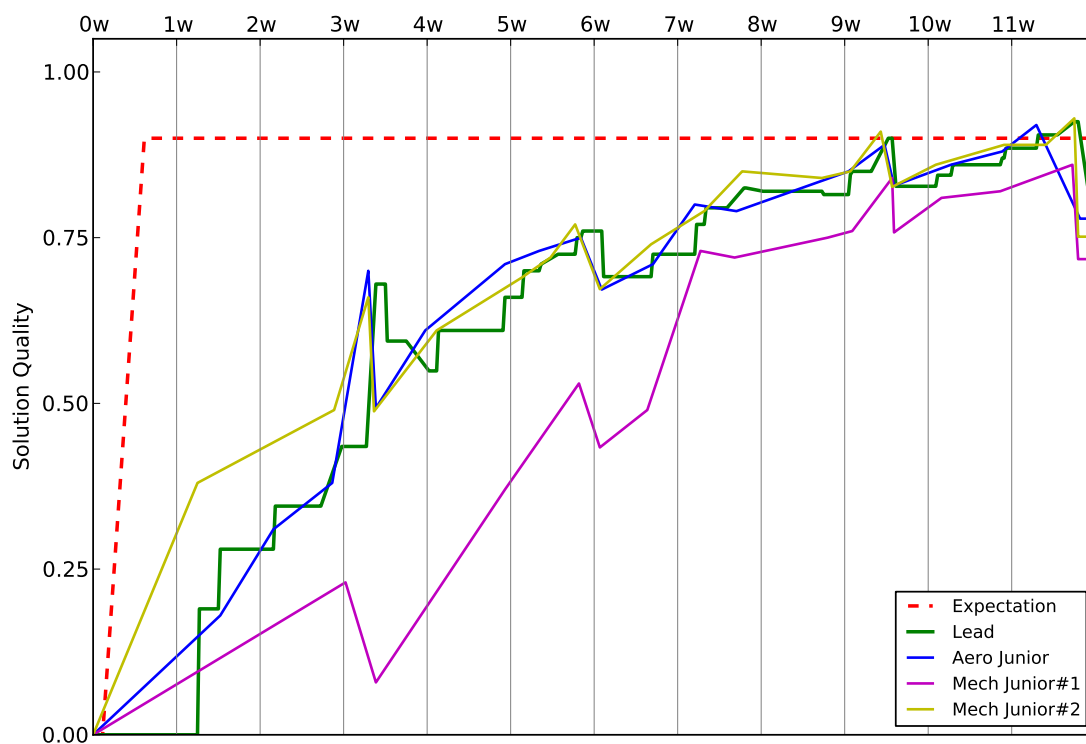


Figure 6.8: Solution quality level evolution during the course of a single simulation run.

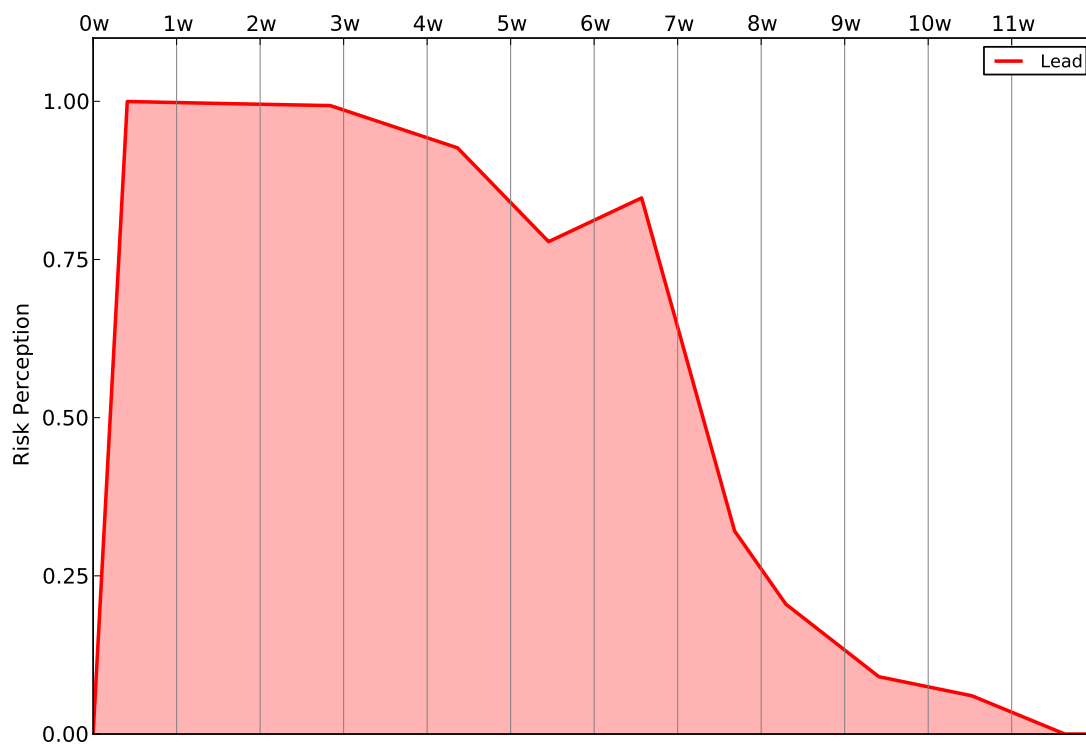


Figure 6.9: Risk perception level evolution during the course of a single simulation run.

evolution shows how the work performed by aerodynamic and mechanical designer agents contributes to the overall development of a collaborative solution for the system aiming to reach the level of quality expected by the Customer agent. The collaborative solution quality level is perceived by the Lead agent according to the weight assigned to each solution component. In this particular setup, equal importance was given to both disciplinary aspects of the design solution, as stated in Section 6.4.1. Figure 6.8 makes also visible the effects of uncertainty in the agent’s design efforts, showing how the quality level suffered a deterioration – the “knock-downs” visible around weeks 3, 6 and 9, for instance – when events of requirements change occurred during simulation.

Figure 6.9 presents the evolution of the level of risk perceived by the Lead agent during the course of the same simulation. As stated in Section 6.3.7, this is an important internal decision-making variable used for resource adjustments and arises from an in-situ evaluation of time left until the deadline and the current gap in quality relative to the Customer’s expectation. Analysis of Figure 6.9 shows that the level of risk perceived by the Lead agent decreases according to the rise in the solution’s quality. It is also relevant to demonstrate how the perceived risk rises again around week 6 following an update of requirements arriving from the Customer. Looking back to the process chart presented Figure 6.6, it is possible to conclude that this re-evaluation of risk at week 6 led the agent to request additional resources to perform design work, namely to both Senior Designers who subsequently provided design support between weeks 6 and 7.

As a result of the resource utilization made by the design team, Figure 6.10 sheds light to the cost evolution during the course of the simulation, showing how both the hourly and cumulative cost consumed by the project evolved during simulation time. Peak values of hourly cost consumed by the project occurred between weeks 3 and 4 and also between weeks 6 and 7, as a result of time periods where all resources available in the supplier organization were performing design work in a *concurrent* manner (visible in Figure 6.6). The accumulated cost incurred by the organization during the course of 12 weeks to execute the project reached approximately 67 thousand cost units<sup>17</sup>.

---

<sup>17</sup>These exploratory simulations shall refer to arbitrary project cost units. The reader can relate to its own monetary units, such as €, £ or \$.

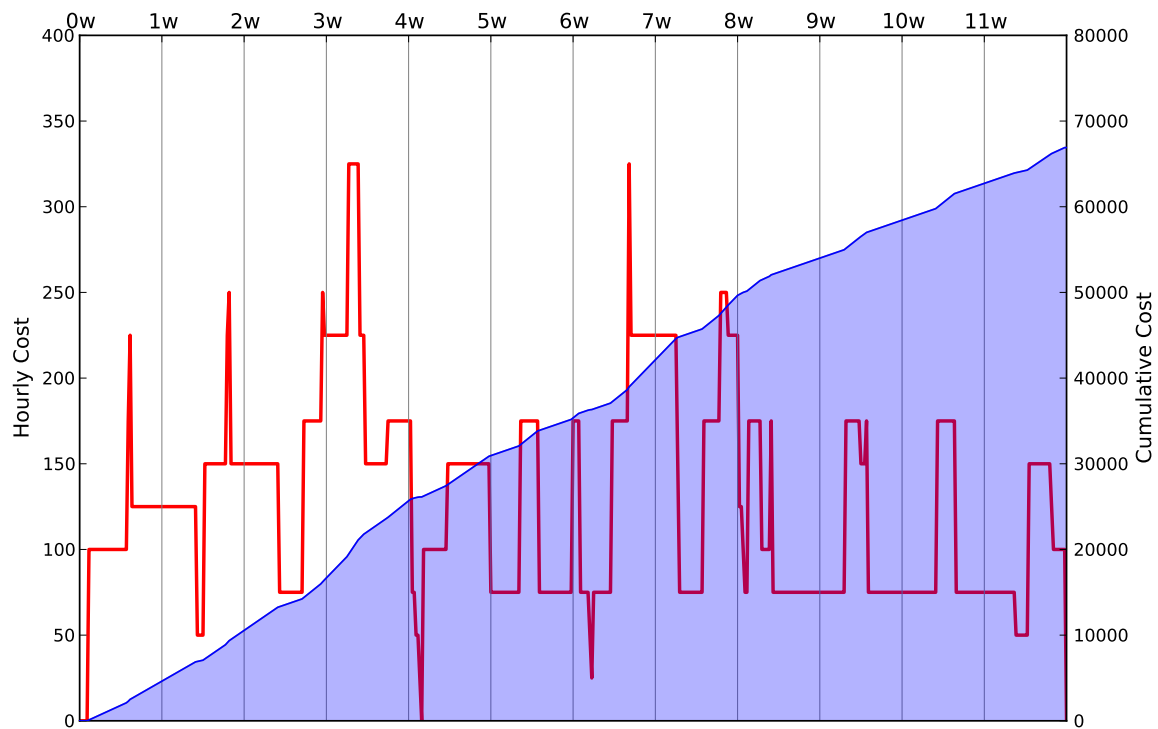


Figure 6.10: Cost evolution during the course of a single simulation run. Hourly and cumulative costs are presented in project cost units.

#### 6.4.4 Project performance evaluation

Considering a simulation run as an individual experiment with the system, performing a large number of simulations is a standard approach to understand and characterize in a statistical manner the effects of variability in the system's performance, which is a technique commonly known as the Monte Carlo method. Due to its importance, multi-run post-processing of AMPERE simulations has also been explored and results are presented in Figures 6.11 and 6.12.

Figure 6.11 depicts two measures of project performance related to the solution quality achieved by the design team and submitted to the Customer: 1) the average quality level of proposals submitted during each project; and 2) the quality level of the last proposal submitted. Frequency distributions of the average and last proposal arise from 50 simulation runs of this simple model created with the setup described in Section 6.4.1. Figure 6.11 compares both measures with the expected quality level that was setup in the Customer's agent. It shows, for instance, that the project was able to meet or exceed the

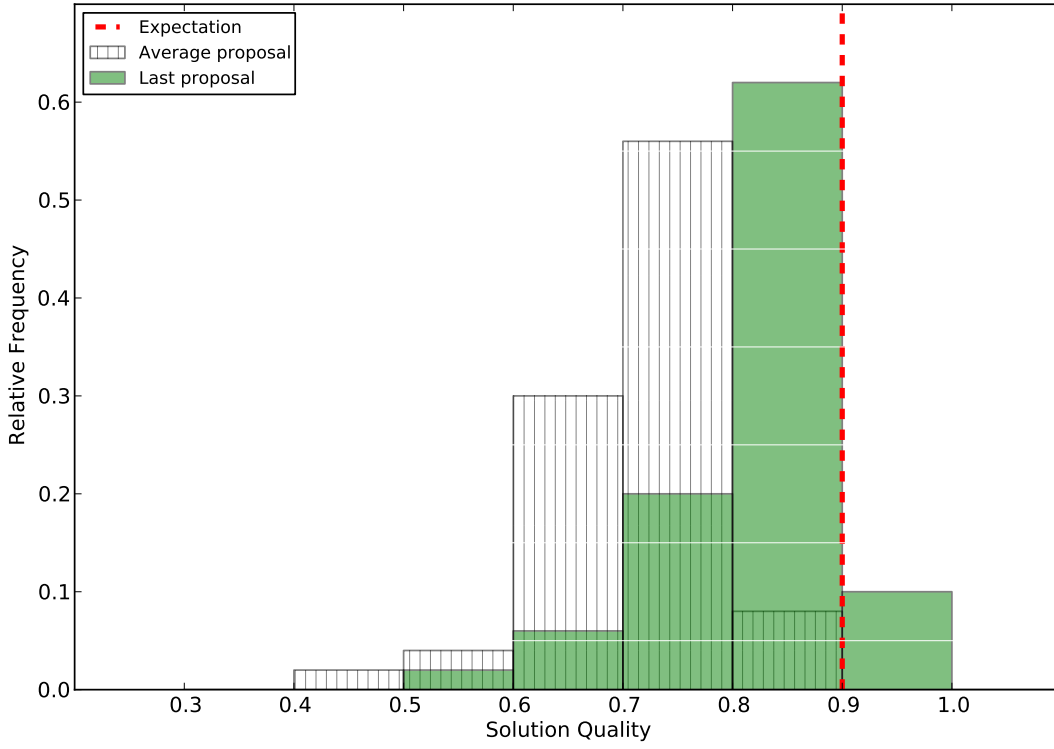


Figure 6.11: Histogram of solution quality from 50 simulation runs.

customer's expectations in approximately 10% of the cases. Adopting a frequentist view of probability, this enables an estimation of the probability of satisfying the Customer with the last proposal issued – in this case,  $P(Q_{lastproposal} \geq Expectation) \approx 0.1$  – for a particular model setup. The mean ( $\mu$ ), median ( $M$ ), standard deviation ( $\sigma$ ) or other statistical metric characterizing the solution quality achieved by the team during multiple simulations provide also ways to evaluate project performance. In these particular simulations, the team attained a quality in its last proposal characterized by  $\mu = 0.83$  and  $M = 0.85$ .

Moreover, Figure 6.12 shows the frequency distribution of cumulative costs incurred by the projects during the same number of simulations. It thus provides a second dimension for project performance evaluation. Considering that organizations normally have budget constraints, as illustrated in Figure 6.12, analysis of results from multiple AMPERE simulations provide also ways to understand the probability of meeting or exceeding the budget available to perform preliminary design projects. In this case, the probability of conforming with the target budget was found to be approximately 70 % from Figure 6.12,

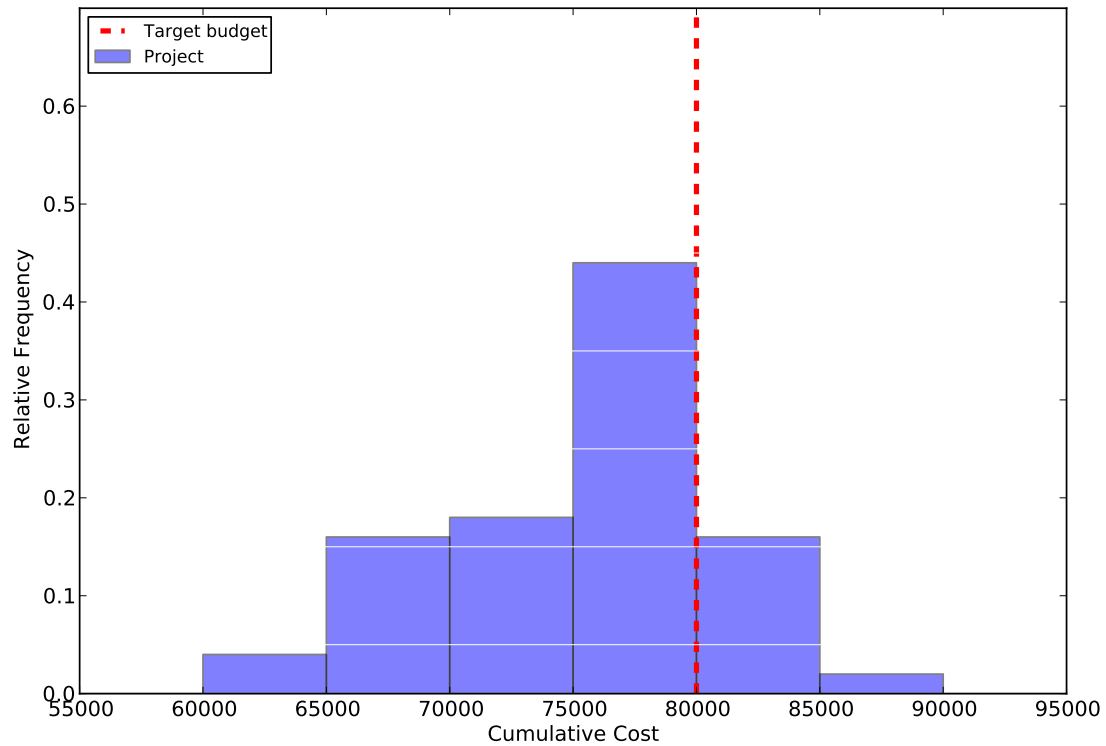


Figure 6.12: Histogram of cumulative project cost from 50 simulation runs.

i.e.,  $P(C_{project} \leq Target) \approx 0.7$ . For the simulation setup described in Section 6.4.1, statistical treatment of 50 simulation runs resulted in a mean ( $\mu$ ) and median ( $M$ ) project cost of respectively 75.6 and 76 thousand cost units.

### 6.4.5 Effects of change in performance

Recalling from Chapter 1 that this thesis argued in favor of the use of model-based simulations to investigate complex cause-effect relationships, exploratory simulations have also been used to perform an initial investigation of the relationship between change and its effects in project performance.

This investigation is presented in Figure 6.13. The study consisted essentially in the variation of the external environment's Time between Changes (TbC) setup interval – keeping all other setup variables constant – and observe the response behaviour arising from simulation. The author departed from the initial setup defined in Section 6.4.1 ( $TbC = Unif(10, 20)$ ) and varied the bounds of the uniform distribution defining the

TbC in steps of 5 working days, i.e., one working week. Similarly to the process followed in Section 6.4.4, 50 simulation runs of the model were performed with each modified setup and the project's performance was measured in terms of two key metrics: the solution quality achieved in the last proposal delivered by the design team; and the cumulative cost incurred by the project. Both were characterized statistically for each set of 50 simulation runs using the median of quality and cost values.

The study's results appear in Figure 6.13 together with a third order polynomial curve fit to the data points found from simulation. Figure 6.13 reveals an interesting solution quality response behaviour. It becomes apparent that there is an interval or "plateau" – between 3 and 5 weeks – where variations in the Time between Changes arising from the external environment have little effect in the solution quality delivered to the Customer. In addition, it shows that further reductions in the Time between Changes below 3 weeks produces a reduction of the last proposal's solution quality with increasing rates (Figure 6.13). On the other hand, project cost appears to behave rather linearly and inversely to the increase in the Time between Changes. The author argues that this is intuitive since designer agents spent less time looking for a new design solution due to updated requirements and thus progress faster as the rate of arrival of changes decreases.

In addition, Figure 6.13 supports this thesis arguing that the previous response behaviours suggest that design systems possess a *stability region* relative to the arrival rate of changes and an *instability point*, which determines the transition to a state where changes arrive faster than what the system can cope with. Such state should naturally be avoided since project performance is significantly affected. This is a major hypothesis that requires further research through the development of more complex – and realistic – models of early design, such as those where the supplier may be composed of several hierarchically organized design teams, each responsible for a sub-system or component and capable of communicating changes independently and at a particular frequency to other teams.

However, although these results arise from a very simple model, exploratory simulations have shed light to how agent models such as AMPERE may be used to investigate the effects of change in project performance with the purpose of generating novel academic

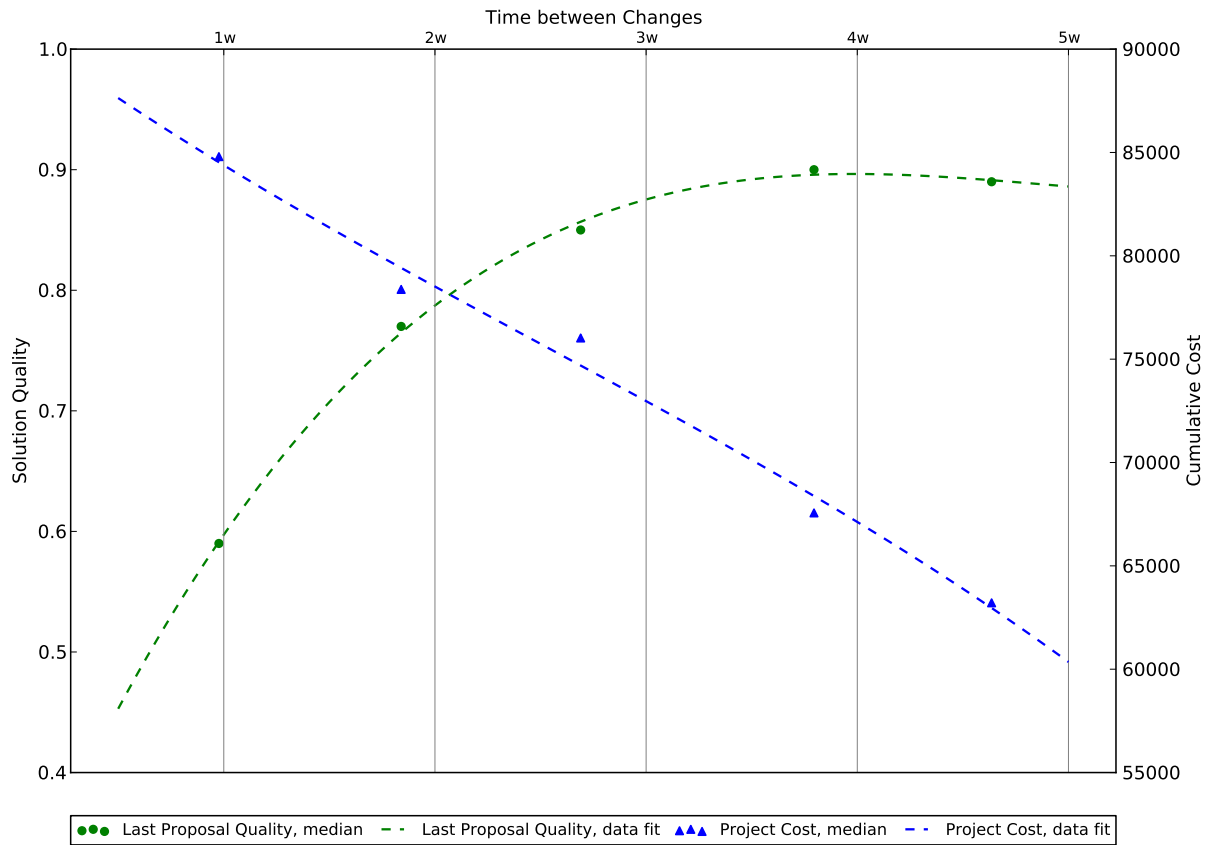


Figure 6.13: Study of the effects of change in project performance. Each data point arises from 50 simulation runs of a particular environment setup with a modified Time between Changes. Performance data refers to median values.

knowledge about the behaviour of design systems or with the aim of supporting industrial decision-makers managing design processes and planning projects for change.

### 6.4.6 Discussion

The previous sections explored the outputs arising from a simple AMPERE model that the author created to conceptualize an early design context where a single customer engaged with a supplier comprising only one design team. Even with such a simple model of early design, this dissertation argues that exploratory simulations have demonstrated the potential of the agent model in three different research dimensions.

The first has been the confirmation that agent-based models are an appropriate and promising approach to capture the dynamics of early design, which is when requirements change occurs more often and planning is most difficult. Looking back to results presented



in Sections 6.4.2 and 6.4.3 – which showed the visualization of the design process followed by agents and the evolution of agent internal variables during simulation – the author argues that this research demonstrated that agent-models allow capturing the effects of uncertainty and change, various facets of iteration, concurrency of activity streams, frequent social interactions and information exchange between teams, collaborative design efforts, and distributed and adaptive decision-making in response both to organizational commitments and to the occurrence of unforeseen events.

The second dimension shown by these exploratory simulations is the potential of the AMPERE framework to support industrial organizations understanding and evaluating the most likely performance of projects during early design, for a given level of change that may be expectable. Results presented in Section 6.4.4 showed how the analysis of a considerable number of simulations provides an estimate of the likely level of solution quality achieved and proposed to the Customer and also an estimate of the likely cost that will be consumed by the project to accomplish a certain proposal quality level. Furthermore, the outputs of exploratory simulations demonstrated also the potential for evaluating the probability of the organization satisfying the Customer’s expectations or the budget available to perform the project. These are key performance metrics which may be used within industrial organizations during project planning activities to make more informed decisions about the resources or budget to be made available to design teams. The author thus argues that initial simulations have clarified how project performance may be assessed and how AMPERE may support planning for an expectable level of change during early design stages.

Lastly, Section 6.4.5 has shed light to the potential use of AMPERE to support systematic investigations to the effects of change in project performance through the realization of sensitivity studies such as the one presented in Figure 6.13, which allow the derivation of response curves of the system for varying levels of change. Section 6.4.5 has revealed also the potential for further research using the AMPERE framework in order to determine points of instability regarding the arrival of change from model-based simulation of complex product development processes. Such studies can potentially support industrial

stakeholder performing management and planning decisions.

In addition, the author's belief is also that these exploratory simulations have opened two main avenues for further academic research. The first relates to further development and application of agent-models using the AMPERE framework to early design process management problems. An expert in agent technologies and agent-based modelling shall recognize that the initial model created by the author contains several simplifications which may become limitations in future applications.

For instance, the Belief-Desire-Intention reasoning engine implemented is similar to an automatic planning engine where the agent goes through a pre-compiled library in a sequential manner to choose the best course of action and executes it. The agents do not possess the capability to reconsider its intentions or measure its success based on updated perceptions of the environment's status. In the current model, the Lead agent instructs Design agents to continue working until the deadline regardless of the perceived likelihood of meeting the Customer agent's expectations. This an example of an unrealistic behaviour that could be removed through the introduction of reconsideration capabilities during planning processes. Further research and developments in the AMPERE framework are thus needed in order to support industrial organizations understanding and solving design process management problems through agent-based models.

The second – and most ambitious – research avenue foreseen by the author is the development of a novel software platform based in AMPERE for design process modelling using agent technology which can be made available for the broader use of the academic and industrial product development community.

As noted in Chapter 5, lack of cost-effectiveness and visualization power of agent-models have so far limited its penetration in industrial practices. However, the research presented in this chapter has demonstrated the current agent development platforms – such as SPADE – allow the development of customized models with far less effort than in the past. Because of that, it is critical that further research makes available to the design community the use of agent-based modelling with decreasing effort. The design and development of this software package to fulfill this vision is, by itself, a large undertaking

which will require significant research efforts. However, the author believes it is a key step to facilitate the penetration of agent-based modelling in design organizations for planning and research purposes.

## 6.5 Research progresses

This chapter presented and explored an agent-model for planning and research of early design, which has been developed by the author with the purpose of addressing the remaining set of questions – Q3, Q3-b and Q3-c – included in the third group of research questions, previously defined in Section 1.4. The research progresses made throughout the course of this chapter allowed the author to answer these questions in the following manner:

**Q3** *How can model-based approaches support planning for expectable levels of change during early stages of complex product development projects?*

Based on a general-purpose agent development platform, the author synthesized an Agent Model for Planning and rEsearch of eaRly dEsign (AMPERE) to support planning for change during early complex product development. This model has conceptualized key actors of early design stages, such as an external Customer agent, a Design Lead agent and Junior and Senior Designer agents belonging to a design team. These agents have been realized with a specific set of possible agent actions arising from prior empirical research of early design stages made by the author at a major gas turbine manufacturer. During simulation, the design agent's actions are highly influenced by the external stakeholder's requests and changes of requirements arising from the environment. Results from AMPERE simulations include the level of solution quality achieved by the design team and the cost associated, which are key performance metrics supporting planning activities.

**Q3-b** *What approach is appropriate to capture the dynamics of early design, which is when requirements are changing often?*

Among the different types of product development modelling approaches, agent-based models offer the most appropriate set of capabilities to capture key dynamics of early

design stages, such as uncertainty, iteration, collaboration and adaptation. Results from AMPERE simulations reported in this dissertation showed that agent-models facilitate the capture of concurrency in activity streams, frequent social interactions and asynchronous information exchange between design teams, collaborative efforts and distributed and adaptive decision-making of design actors. These are key features which have been found necessary through empirical research to model early design stages of complex product development.

**Q3-c** *How can model-based approaches support the estimation of project performance during early design stages taking into account the level of expectable change?*

For a given level of expectable change, AMPERE simulations allow the estimation of the likely level of solution quality achieved and the project cost that will be consumed to accomplish a certain proposal quality level. In addition, the agent model supports also the estimation of the probability of satisfying external stakeholder expectations or the internal budget that can be made available for project activities. These are key performance dimensions which may be used by industrial organizations during early project planning activities. In addition, AMPERE supports also the realization of sensitivity studies to understand and characterize the project's response to varying levels of incoming change.

## **6.6 Summary**

To recapitulate, the motivation for this chapter arose from the need to bring forward novel tools and approaches supporting engineers planning complex product development projects during early design stages, which is when the effects of change are difficult to predict and consequently the task of planning project resources and budget is most challenging.

Drawing from the comparative review of strengths and limitations of various product development modelling and simulation approaches, this chapter argued that agent-models offer the most promising set of capabilities among the approaches available to capture the dynamics of early design, including a superior representation of organizational features

and social interactions and the benefits of decentralized control, which decreases the development effort and ensures greater scalability compared to other approaches.

Based on the choice of an agent-based approach, the chapter reported the development of the Agent Model for Planning and rEsearch of eaRly dEsign (AMPERE) based on the utilization of the Smart Python multi-Agent Development Environment, an agent development technology relying on standards defined by the Foundation for Intelligent Physical Agents. The vision for model development included the representation of organizational structures, design process activities, the effects of uncertainty, iteration, design collaboration and adaptive decision-making behaviour under the framework and the prediction of the quality and cost of the agent's work as a result of the changes arriving from the environment.

The chapter described the architectural design of AMPERE, the definition of agents and their internal behaviours and social interactions. The model included the realization of several types of agents found during early design stages – such as the Customer, the design Lead and Junior and Senior Designers – based on an internal practical reasoning algorithm. Following standard agent-based modelling methodologies, each agent type was equipped with a finite set of actions available and the conditions under which they can be performed on the environment that the agent is allowed to reach. The agent actions, including the ability to interact with other agents, were defined according to empirical research performed by the author with the purpose of capturing the dynamics of early design.

Upon the completion of the agent-model, an initial set of exploratory simulations were performed and also reported in this chapter. These simulations were based on a simple model of early design: a single Customer agent that engaged regularly with a single supplier organization, where one design team composed of a Lead agent and five Designer agents worked to satisfy the Customer's requests and adapted its actions to the arrival of changes in requirements. The chapter showed the visualization of the design process followed by agents during simulation, the time evolution of key agent internal variables and the evaluation of the project's performance based on the solution quality delivered

and the cumulative cost incurred. In addition, it reported also an investigation to effects of varying levels of expectable change on the project's performance metrics.

## Part IV

# Conclusions and Future Work

If I have seen further it is by standing  
on the shoulders of giants.

—ISAAC NEWTON





# Chapter 7

## Conclusions

The current chapter concludes this dissertation. It begins with a summary of the progresses made regarding the three groups of research questions identified in Chapter 1. The key contributions to academic knowledge and to Rolls-Royce stakeholders arising from the research reported in this dissertation are subsequently reviewed and discussed. Taking into consideration the different research contributions, this chapter sheds light to opportunities for further research.

### 7.1 Progresses made on research questions

This section provides an overview and summary of the progresses made regarding this dissertation's research questions.

#### 7.1.1 Understanding causes

This thesis's first set of research questions targeted at understanding how requirements evolve and change during complex product development projects, what are the best requirements management practices, what are the root causes of change in practice, how they vary over time and how much change arises from external and internal factors. To recapitulate, the research reported in Chapters 2 and 3 allowed this dissertation to progress regarding all these questions and to provide the following set of key conclusions:

**Q1** *How do requirements evolve and change during complex product development projects, such as the ones executed by Rolls-Royce?*

Case-study findings support the conclusion that requirements evolve and change in organizations developing complex technical systems through modifications, additions and removals. This dissertation showed that the number of requirements at a particular system layer may double during the course of the development process and may only stabilize in late stages of complex product development projects, already during prototype development and testing.

**Q1-a** *What are the best practices in requirements management?*

This thesis outlined that best requirements management practices include a core engineering process and support change management activities. The core process encompasses the elicitation of requirements from stakeholders, analysis and negotiation, followed by requirements documentation and validation. The change management process supports the core engineering process with analysis of problem reports and change requests, assessment of the impact of change and implementation of a selected solution leading to a revised version of requirements.

**Q1-b** *What are the root causes of requirements change during the design of large technical systems?*

Empirical findings demonstrated there is a wide range of causes of requirements change during the development of large technical systems. Some change arises from high-level stakeholders, such as the customer, certification authorities or internal business functions. Various elicitation related problems lead to change due to missing requirements, incorrect or incomplete capture, ambiguous language and due to redundancy or merger of requirements that had been already captured. Difficulties surrounding the requirements cascade process originate change due to incorrect or incomplete flow down. And the update of traceability information across the document landscape is also a root cause of change.

**Q1-c** *How do the causes of change vary during different phases of the product develop-*

*ment process?*

This dissertation presented case-study results supporting that the root causes of change vary significantly during different phases of the development process. Empirical data showed that the early development stages were mainly governed by requirements modifications due to incomplete elicitation, while the late development stages were dominated by incomplete flow down of higher-level specifications that had undertaken a long and laborious negotiation process between supplier and customer representatives.

**Q1-d** *How much change is externally and internally driven?*

It was demonstrated that more than 80% of changes generated during the development of the complex system investigated in this dissertation had internal root causes and the amount of change generated from the customer remained relatively constant during the development process, around 15%.

## **7.1.2 Managing uncertainty**

The second set of research questions aimed to clarify the main types of uncertainty affecting design activities, to support engineers and designers comprehending and handling uncertainty in the design information exchanged during collaborative and distributed engineering processes and investigating how does the level of uncertainty in key design variables such as requirements varies during product development and what is the typical level that should be expected by designers engaged in new projects. To recapitulate, the research presented in Chapter 4 allowed this dissertation to make progresses in the previous research questions and to bring forward the following set of key conclusions:

**Q2** *How can the level of uncertainty in requirements be predicted and managed in industrial practice?*

This dissertation proposed a structured and practical approach – coined The Imprecision Management Method – supporting designers and engineers predicting and managing the level of uncertainty in key design information such as requirements

based on the realization of five steps: 1) collection of historical records about uncertain variables from past projects; 2) reconstruction of the design variables' time evolution using the records collected; 3) statistical characterization of design imprecision based on two key indicators, the Magnitude of Change and the Time between Changes; 4) communication of imprecision levels to new projects based on a Matrix of Imprecision, which relates the average magnitude and arrival rate of changes; and 5) continuous knowledge update from new projects to ensure increasing accuracy of the method.

**Q2-a** *What are the main types of uncertainty in engineering design?*

The research showed that variability or aleatory uncertainty, model uncertainty and design imprecision are three key types of uncertainty affecting the activities of designers and engineers during collaborative and distributed engineering processes. Variability or aleatory uncertainty denotes the amount of variation expected in a design variable as a result of random or uncontrolled processes. Model uncertainty is used to refer to variations in the values computed from engineering models due to mathematical, physical or numerical approximations. And design imprecision was found in prior literature to refer to variations in the value assigned to variables due to evolving preferences of engineers and designers during the design process.

**Q2-b** *How does the level of uncertainty in requirements and design variables varies during the course of complex product development projects?*

Findings arising from an industrial case-study investigating the applicability of the Imprecision Management Method revealed that the level of uncertainty in requirements or other variables varies considerably during complex product development. It was shown that uncertainty levels were much higher during the first 2 years of product development, which corresponded roughly to the preliminary design stage. Empirical results from the case-study revealed that functional requirements flown down to sub-system design teams evolved from an average Magnitude of Change lower or equal to 4% during the first 6 months of the project to 1% approximately 24 months after the project's start. It was also found that the average Time between

Changes evolved from approximately 4.6 weeks to 14.1 weeks during the same time period of the projects.

**Q2-c** *What is the typical level of uncertainty that should be expected by designers and engineers in new projects?*

This dissertation found that most of the requirements communicated regularly to sub-system designers can be expected to have an average Magnitude of Change below 4% and change with an average Time between Changes of 7 weeks during the preliminary design of new product development projects. However, the case-study findings also suggested that a particular subset within the group of requirements studied is highly uncertain and magnitudes of change between 10% and 20% should be expected by sub-system engineers and designers.

### 7.1.3 Planning for change

Lastly, this thesis' third set of research questions targeted at investigating what approaches are available to model complex product development, how they can capture the dynamics of complex product development during early design stages and aimed to investigate the effects of expectable levels of requirements change in project performance variables together with novel approaches supporting planning during early project stages. To recapitulate, the research presented in Chapters 5 and 6 allowed this dissertation to make progresses in these questions and to provide the following conclusions:

**Q3** *How can model-based approaches support planning for expectable levels of change during early stages of complex product development projects?*

This dissertation synthesized an Agent Model for Planning and rEsearch of eaRly dEsign (AMPERE) to support planning for change during early complex product development. The model includes key actors of early design, such as an external Customer agent, a Design Lead agent and Junior and Senior Designer agents belonging to a design team. These agents have been realized with a specific set of possible agent actions arising from prior empirical research of early design made

by the author. During simulation, the design agent's actions are highly influenced by the external stakeholder's requests and changes of requirements arising from the environment, which influence the level of solution quality achieved and the cost incurred by the team. Quality and cost are key performance metrics supporting planning activities.

**Q3-a** *What approaches are available to model complex and uncertain design processes? What are their main strengths and limitations?*

This dissertation showed that the main approaches include activity-based models and its variants, agent-based models and system dynamics models. It was found that the main strength of activity-based models relying on precedence and dependency relationships among activities is their ability to represent in a cost-effective manner well-structured processes, such as those typical of the detailed design project stages. Conversely, activity-based models relying on adaptive algorithms and agent-based models were found to be more equipped to represent ill-defined processes where there are concurrent, frequent and collaborative interactions between participants and the adaptive behaviour of design teams is often observed, which are characteristics typical of the concept and preliminary design stages of complex development projects. The strengths of adaptive models and agent-based models come at the expense of visualization and cost-effectiveness to support planning.

**Q3-b** *What approach is appropriate to capture the dynamics of early design, which is when requirements are changing often?*

Findings from AMPERE development and simulation showed that agent-based models offer the most appropriate set of capabilities to capture key dynamics of early design stages. Simulation results reported in this dissertation demonstrated that agent-models facilitate the capture of concurrency in activity streams, frequent social interactions and asynchronous information exchange between design teams, collaborative efforts and distributed and adaptive decision-making of design actors, which are key characteristics found in early design.

**Q3-c** *How can model-based approaches support the estimation of project performance during early design stages taking into account the level of expectable change?*

AMPERE simulations foster the estimation of the likely solution quality and cost from design work, for a given level of expectable change arising from the environment. The agent model supports also the estimation of the probability of satisfying external stakeholder expectations or conforming with the internal budget that can be made available for project activities. Sensitivity studies to understand and characterize the project's response to varying levels of incoming change are also available to support planning for change.

## 7.2 Review of research contributions

This dissertation was motivated by the needs to understand the causes of requirements change, predict and manage uncertainty levels and to develop model-based approaches to support planning for change during the early design stages of complex product development. During the course of the research project performed to answer the previous research questions, a number of key research contributions to academic research and to Rolls-Royce stakeholders were realized:

- **Contributions to systems engineering and requirements management.**

This dissertation contributed with a large-scale empirical study performed at Rolls-Royce which allowed to gain an increased understanding about the requirements evolution and the causes of change at different phases of the product development process of complex technical systems. This research generated original quantitative and qualitative findings about requirements change during complex product development processes which constitute academic contributions complementing previous empirical studies – e.g. Almefelt et al. (2006); Sudin and Ahmed (2009); Vianello and Ahmed-Kristensen (2012) – and recognized literature – e.g. Blanchard and Fabrycky (2006); Pohl (2010) – concerned with systems engineering and requirements management. Moreover, various management guidelines aiming to improve require-

ments management practices have also emerged from this research for Rolls-Royce stakeholders.

- **Contributions to uncertainty management in engineering design.** This dissertation contributed with the synthesis of the Imprecision Management Method, a method aiming to support large organizations understanding, quantifying and communicating uncertainty levels during complex product development, particularly uncertainty in key variables such as requirement which are often used as inputs to the activities of design teams. In addition, this thesis contributed with a case-study which revealed to academia and to Rolls-Royce how the level of uncertainty in requirements evolve and what is the typical level that should be expected in new product development projects. The research showed that the Imprecision Management Method is a simple and practical approach that can communicate uncertainty within organizations through indicators familiar to practitioners and promote a consistent understanding of its meaning within design teams. In the context of the work of Wood et al. (1990) and Otto and Antonsson (1994), these contributions complement and extend prior research on uncertainty quantification and management in engineering design.
- **Contributions to complex product development modelling.** This dissertation contributed with the synthesis of the Agent Model for Planning and rEsearch of eaRly dEsign, which is both an agent model and a scalable modelling framework aiming to support design organizations planning for change during early design stages of complex product development. Results from exploratory simulations based on a simple model showed that the agent model developed by the author can be used as a tool supporting engineers in project planning – through the estimation of likely quality and cost arising from project activities – as well as to research the effects of change in project performance and discover points of instability. The Agent Model for Planning and rEsearch of eaRly dEsign and the results reported in this dissertation are original research contributions which extend prior literature on the topic of product development modelling and simulation.



## 7.3 Opportunities for future research

Analysis and reflection over the previous research contributions have led the author to identify several opportunities for future research arising from this dissertation. These opportunities are:

(i) **Empirical research on the impact/ cost of requirements change.**

Chapter 3 analyzed and discussed the frequency of causes of change arising from the empirical study against the potential impact/ cost of each change implementation. The latter was evaluated qualitatively according to the author's insights acquired during field research at Rolls-Royce. This led to the hypothesis that most of the cost incurred by organizations due to changes in requirements results from a relatively small proportion of the amount of change that is actually performed during complex development projects. Performing further empirical investigation to quantify and validate the actual impact/ cost of each type of requirements change and assess with scientific rigor the influence of each type in the total cost of change for projects is thus a major opportunity for future research arising from the findings reported in Chapter 3.

(ii) **Industrial applications of the Imprecision Management Method.**

Chapter 4 demonstrated that it is feasible to recover historical knowledge from project archives and use it to understand, characterize and communicate uncertainty in industrial organizations developing large technical systems. This suggested that the Imprecision Management Method has industrial value and may be applied in practice. There is thus an opportunity for research arising from the actual implementation of the method in projects at Rolls-Royce or in other industrial organizations which is to evaluate and measure the benefits arising from the method's utilization during actual design process management. Further research questions include the investigation of the decision-making behaviour of designers when they are in possession of the expected amount and frequency of change and how this information affects the effectiveness of design process management.

(iii) **Integration of the Imprecision Management Method with other approaches.**

Chapter 4 showed that the Imprecision Management Method provides historical data about the frequency and magnitude of change. It argued also that historical records supporting these change indicators could be used to estimate or model the probability and the impact of a change. Considering that prior change propagation methods reported by Clarkson et al. (2004) or Giffin et al. (2009) rely on estimations of the probability and impact of change, there is thus the opportunity of integrating the Imprecision Management Method and multidomain change propagation methods, which can open new avenues of research within the topic of change management.

(iv) **Industrial applications of the Agent Model for Planning and rEsearch of eaRly dESign.**

Chapter 6 demonstrated that one avenue for further research using the Agent Model for Planning and rEsearch of eaRly dESign framework is to determine points of instability regarding the arrival of change in complex product development processes and the development of more complex agent models representative of realistic industrial environment through additional empirical research at industrial companies such as Rolls-Royce. Both thus relate to the application of the agent-modelling framework in industry to understand and support resolution of specific design management problems.

(v) **Development of a general software platform for design process modelling using agents.**

Chapter 6 identified also the development of a general-purpose software platform for design process modelling using agent technologies as a major avenue for further research based on the initial agent modelling work reported in this dissertation. Chapter 5 pointed out that lack of cost-effectiveness and visualization power of agent-models have so far limited its introduction in industrial organizations. The research performed by the author during the development of the Agent Model for Planning and rEsearch of eaRly dESign suggests that these problems can be overcome. There is thus a major research opportunity consisting of the design and

development of a large software package to be made available to the academic and industrial community for design process modelling using agent technologies.

## **7.4 Conclusion**

Requirements change is challenging for organizations developing large technical systems. This dissertation has delivered a large-scale empirical study which allowed to understand the requirements evolution and the causes of change at different stages of the development process. In addition, a novel method aiming to support designers and engineers predicting and managing uncertainty levels has been proposed and its applicability in the industrial environment has been evaluated through an initial case-study. Lastly, this dissertation has developed an original agent model to support planning and research of early design stages taking into account the level of expectable change. The contributions arising from the research reported in this thesis have in turn led to several promising directions for future research.



# Bibliography

- B. Adelson. Developing strategic alliances: a framework for collaborative negotiation in design. *Research in Engineering Design*, 11(3):133–144, 1999.
- L. Almfelt, F. Berglund, P. Nilsson, and J. Malmqvist. Requirements management in practice: findings from an empirical study in the automotive industry. *Research in Engineering Design*, 17:113134, 2006.
- G. Altschuller. *Erfinden: Wege zur Lösung technischer Probleme*. VEB Verlag, Berlin, 1984.
- M. Andreasen. *Machine design methods based on a systematic approach - contribution to a design theory*. PhD thesis, Department of Machine Design, Lund Institute of Technology, Lund, Sweden, 1980.
- E. K. Antonsson and K. N. Otto. Imprecision in engineering design. *Journal of Mechanical Design*, 117(B):25–32, 1995.
- AnyLogic. Anylogic home page. <http://www.anylogic.com/>, 2013.
- P. D. Arendt, D. W. Appley, and W. Chen. Quantification of model uncertainty: calibration, model discrepancy, and identifiability. *Journal of Mechanical Design*, 134(10):100908, 2012.
- M. Asimow. *Introduction to design*. Prentice Hall, 1962.
- J. M. Aughenbaugh and C. J. Paredis. The value of using imprecise probabilities in engineering design. *Journal of Mechanical Design*, 128(4):969–979, 2006.

- R. Axelrod. *The complexity of cooperation: agent-based models of competition and collaboration*. Princeton University Press, 1997.
- O. Barron, O. Kim, S. Lim, and D. Stevens. Using analysts' forecasts to measure properties of analysts' information environment. *Accounting Review*, 73(4):421-433, 1998.
- C. Bell, D. C. Wynn, W. N. Nawes, and P. J. Clarkson. Using meta-data to enhance process simulation and identify improvements. In *Proceedings of the 16th International Conference in Engineering Design (ICED07)*, Paris, France, August 2007.
- F. Bellifemine, A. Poggi, and G. Rimassa. Developing multi-agent systems with jade. In C. Castelfranchi and Y. Lesprance, editors, *Intelligent Agents VII Agent Theories Architectures and Languages*, volume 1986 of *Lecture Notes in Computer Science*, pages 89–103. Springer Berlin Heidelberg, 2001.
- D. Bergsjö, L. Almfelt, and J. Malmqvist. Supporting requirements management in embedded systems development in a lean-influenced organization. In *Proceedings of the 11th International Design Conference (DESIGN 2010)*, pages 1025–1034, Dubrovnik, Croatia, May 2010.
- L. Bertalanffy. *General systems theory, foundations, development, applications*. George Braziller Inc., New York, 1968.
- B. S. Blanchard and W. J. Fabrycky. *Systems engineering and analysis*. Prentice Hall, 4th edition, 2006.
- L. T. M. Blessing and A. Chakrabarti. *DRM, a Design Research Methodology*. Springer Publishing Company, Incorporated, first edition, 2009.
- M. Bratman, D. Isreal, and M. Pollack. Plans and resource-bounded practical reasoning. *Computational Intelligence*, 4(4):349–355, 1998.
- J. Brown. Managing product relationships: enabling iteration and innovation in design. Technical Report June, Alberdeen Group, Boston, Massachusetts, 2006.

- T. Browning. Applying the design structure matrix to system decomposition and integration problems: a review and new directions. *Engineering Management, IEEE Transactions on*, 48(3):292–306, Aug 2001.
- T. R. Browning and R. V. Ramasesh. A survey of activity network-based process models for managing product development projects. *Production and Operations Management*, 16(2):217–240, 2007.
- CAM. Cambridge advanced modeller home page. <http://www-edc.eng.cam.ac.uk/cam/>, 2013.
- L. Cao and S. S. Rao. Optimum design of mechanical systems involving interval parameters. *Journal of Mechanical Design*, 124(3):465–472, 2002.
- M. Carrascosa, S. Eppinger, and D. Whitney. Using the design structure matrix to estimate product development time. In *ASME Design Engineering Technical Conferences (Design Automation Conference)*, pages 1–10, 1998.
- M. P. Case and S. C.-Y. Lu. Discourse model for collaborative design. *Computer-Aided Design*, 28(5):333–345, 1996.
- A. Ćatić and J. Malmqvist. Requirements management when introducing new mechatronic sub-systems - managing the knowledge gaps. In *Proceedings of the 11th International Design Conference (DESIGN 2010)*, pages 661–672, Dubrovnik, Croatia, May 2010.
- A. Chakrabarti, S. Morgenstern, and H. Knaab. Identification and application of requirements and their impact on the design process: a protocol study. *Research in Engineering Design*, 15(1):2239, 2004.
- P. Checkland. *Systems thinking, systems practice*. Wiley, Chichester, 1981.
- P. Checkland. *Soft systems methodology: 30-year retrospective*. Wiley, Chichester, 1999.
- W. Chen, J. K. Allen, K. L. Tsui, and F. Mistree. A procedure for robust design: minimizing variations caused by noise factors and control factors. *Journal of Mechanical Design*, 118(4):478–485, 1996.

- S. Cho and S. Eppinger. Product development process modeling using advanced simulation. In *ASME Design Engineering Technical Conference Proceedings, DETC01/DTM*, page 110, Pittsburgh, Pennsylvania, June 2001.
- M. Christel and K. Kang. Issues in requirements elicitation. Technical Report CMU/SEI-92-TR-012, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, September 1992. URL <http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=12553>.
- J. P. Clarkson, C. Simons, and C. Eckert. Predicting change propagation in complex design. In *Proceedings of ASME design engineering technical conferences*, pages 155–164, Pittsburgh, Pennsylvania, USA,, 2001.
- P. J. Clarkson and J. R. Hamilton. signposting, a parameter-driven task-based model of the design process. *Research in Engineering Design*, 12(1):18–38, 2000.
- P. J. Clarkson, C. Simons, and C. Eckert. Predicting change propagation in complex design. *Journal of Mechanical Design*, 126(5):788–797, 2004.
- K. Cooper. The rework cycle: why projects are mismanaged. *PM network*, pages 174–185, 1993.
- C. A. Cornell. A probability-based structural code. *Journal of American Concrete Institute*, 66(12):974–985, 1969.
- J. Coughlan and R. D. Macredie. Effective communication in requirements elicitation: a comparison of methodologies. *Requirements Engineering*, 7(2):47–60, 2002.
- M. Danilovic and B. Sandkull. The use of dependence structure matrix and domain mapping matrix in managing uncertainty in multiple project situations. *International Journal of Project Management*, 23(3):193 – 203, 2005.
- I. Das. Robustness optimization for constrained nonlinear programming problems. *Engineering Optimization*, 32(5):585–618, 2000.



- R. de Neufville and S. Scholtes. *Flexibility in engineering design*. The MIT Press, Cambridge MA, USA, 1st edition, 2011.
- O. De Weck, C. M. Eckert, and P. J. Clarkson. A classification of uncertainty for early product and system design. In *Proceedings of the 16th International Conference on Engineering Design (ICED'07)*, Paris, France, August 2007.
- A. P. Dempster. A generalization of bayesian inference. *Journal of Royal Statistics Society, Serv. B*, 30:205–247, 1968.
- Q. Dong. *Predicting and managing system interactions at early phase of the product development process*. PhD thesis, Department of Mechanical Engineering, Massachusetts Institute of Technology, Boston, MA, USA, 2002.
- K. Dorst and N. Cross. Creativity in the design process: co-evolution of problemsolution. *Design Studies*, 22(5):425–437, 2001.
- D. Dubois and H. Prade. *Possibility theory*. Plenum press, New York, 1st edition, 1988.
- A. Duffy and F. O'Donnell. A design research approach. In N. Mortensen and J. Sigurjónsson, editors, *Critical enthusiasm - contributions to design science*, page 3340, Tapir, Trondheim, 1999.
- C. Earl, J. Johnson, and C. Eckert. Complexity. In J. Clarkson and C. Eckert, editors, *Design process improvement*, pages 174–197. SpringerVerlag, UK, 2005.
- C. Eckert and P. Clarkson. Planning development processes for complex products. *Research in Engineering Design*, 21(3):153–171, 2010.
- C. Eckert, P. Clarkson, and M. Stacey. The spiral of applied research: a methodological view of integrated design research. In *Proceedings of the 14th International Conference on Engineering Design*, Stockholm, Sweden, August 2003.
- C. Eckert, P. Clarkson, and M. Stacey. The lure of the measurable in design research. In *Proceedings of DESIGN 2004, the 8th International Design Conference*, pages 21–26, Dubrovnik, Croatia, May 2004a.

- C. Eckert, P. Clarkson, and W. Zanker. Change and customisation in complex engineering domains. *Research in Engineering Design*, 15(1):1–21, 2004b.
- M. Eichinger, M. Maurer, and U. Lindemann. Using multiple design structure matrices. In *Proceedings of the 9th International Design Conference*, pages 229–236, Dubrovnik, Croatia, 2006.
- S. Eppinger, D. Whitney, R. Smith, and D. Gebala. A model-based method for organizing tasks in product development. *Research in Engineering Design*, 6(1):1–13, 1994. ISSN 0934-9839.
- S. Eppinger, M. Nukala, and D. Whitney. Generalised models of design iteration using signal flow graphs. *Research in Engineering Design*, 9(2):112–123, 1997.
- J. Fernandes, A. Silva, and E. Henriques. Modeling the impact of requirements change in the design of complex systems. In M. Aiguier, Y. Caseau, D. Krob, and A. Rauzy, editors, *Complex Systems Design & Management*, pages 151–164. Springer Berlin Heidelberg, 2013.
- J. Fernandes, E. Henriques, and A. Silva. A method for managing uncertainty levels in design variables during complex product development. In M. Aiguier, F. Boulanger, D. Krob, and C. Marchal, editors, *Complex Systems Design & Management*, pages 113–124. Springer International Publishing, 2014a.
- J. Fernandes, E. Henriques, A. Silva, and M. Moss. A method for imprecision management in complex product development. *Research in Engineering Design*, 25(4):309–324, 2014b.
- J. Fernandes, E. Henriques, A. Silva, and M. Moss. Requirements change in complex technical systems: an empirical study of root causes. *Research in Engineering Design*, 26(1):37–55, 2015.
- J. Fiksel and F. Hayes-Roth. Computer-aided requirements management. *Concurrent Engineering: Research and Applications*, 1(2):83–92, 1993.

- T. Finin, D. McKay, R. Fritzson, and R. McEntire. Kqml: an information and knowledge exchange protocol. In *Proceedings of the International Conference on Building and Sharing of Very Large-Scale Knowledge Bases*, December 1993.
- FIPA. Agent Communication Language. Technical Report October, Foundation for Intelligent Physical Agents, FIPA 97 Specification Part 2, 1999.
- D. N. Ford and J. D. Sterman. Dynamic modeling of product development processes. *System Dynamics Review*, 14(1):31–68, 1998.
- E. Fricke, B. Gebhard, H. Negele, and E. Igenbergs. Coping with changes: Causes, findings, and strategies. *Systems Engineering*, 3(4):169–179, 2000.
- M. Georgeff and A. Lansky. Reactive reasoning and planning. In *Proceedings of the Sixth National Conference on Artificial Intelligence (AAAI-87)*, pages 677–682, Seattle, WA, US, 1987.
- M. Georgeff, B. P. B, M. Pollack, M. Tamble, and M. Wooldridge. The belief-desire-intent model of agency. In *In Intelligent Agents V*, vol. 1555, pages 1–10, Berlin, 1999. Springer.
- J. K. Gershenson and L. a. Stauffer. Assessing the usefulness of a taxonomy of design requirements for manufacturing. *Concurrent Engineering*, 7(2):147–158, 1999a.
- J. K. Gershenson and L. A. Stauffer. A taxonomy for design requirements from corporate customers. *Research in Engineering Design*, 11(2):103–115, 1999b.
- M. Giffin, O. D. Weck, G. Bounova, R. Keller, C. Eckert, and P. J. Clarkson. Change propagation analysis in complex technical systems. *Journal of Mechanical Design*, 131(8):081001, 2009.
- Y. M. Goh, C. A. McMahon, and J. D. Booker. Development and characterization of error functions in design. *Research in Engineering Design*, 18(3):129–148, 2007.

- O. Gotel and A. Finkelstein. An analysis of the requirements traceability problem. In *First IEEE International Conference on Requirements Engineering*, pages 94–101, Los Alamitos, 1994.
- M. E. Gregori, J. P. Cámara, and G. A. Bada. A jabber-based multi-agent system platform. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems*, AAMAS’06, pages 1282–1284, New York, NY, USA, 2006. ACM.
- A. Griffin and J. R. Hauser. The voice of the customer. *Marketing Science*, 12(1), 1993.
- R. F. Hamade, H. A. Artail, and M. Y. Jaber. Learning theory as applied to mechanical cad training of novices. *International Journal of Human-Computer Interaction*, 19(3): 305–322, 2005.
- B. Hamraz, N. H. M. Caldwell, and P. J. Clarkson. A multidomain engineering change propagation model to support uncertainty reduction and risk management in design. *Journal of Mechanical Design*, 134(10):100905, 2012.
- A. Hatchuel and B. Weil. A new approach of innovative design: an introduction to c-k theory. In *Proceedings of the International Conference on Engineering Design (ICED03)*, Stockholm, Sweden, August 2003.
- J. R. Hauser and D. Clausing. The house of quality. *Harvard business review*, 66(3): 63–73, 1988.
- G. Hazelrigg. A framework for decision-based engineering design. *Journal of Mechanical Design*, 120(4):653–658, 1996.
- A. Heathcote, S. Brown, and D. Mewhort. The power law repealed: The case for an exponential law of practice. *Psychonomic Bulletin & Review*, 7(2):185–207, 2000. ISSN 1069-9384.
- F. S. Hillier and G. J. Lieberman. *Introduction to operations research*. McGraw-Hill, 8th edition, 2005.

- I. Hooks and K. A. Farry. *Customer-centered products: creating successful products through smart requirements management*. AMACOM, 1st edition, 2001.
- IEEE. Standard glossary of software engineering terminology. *IEEE Std. 610. 12-1990*, 1990.
- INCOSE. *Systems engineering handbook: a guide for system life cycle processes and activities*. International Council on Systems Engineering, 3rd edition, 2006.
- O. Isaksson, S. Keski-Seppl, and S. D. Eppinger. Evaluation of design process alternatives using signal flow graphs. *Journal of Engineering Design*, 11(3):211–224, 2000.
- J. Jackson. A keyphrase based traceability scheme. In *IEE Colloquium on Tools and Techniques for Maintaining Traceability During Design*, London, 1991.
- T. Jarratt, C. Eckert, and J. P. Clarkson. Development of a product model to support engineering change management. In *Fifth International Symposium on Tools and Methods of Competitive Engineering*, Lausanne, Switzerland, 2004.
- T. W. Jarratt, C. M. Eckert, N. H. M. Caldwell, and P. J. Clarkson. Engineering change: an overview and perspective on the literature. *Research in Engineering Design*, 22(2): 103–124, 2011.
- J. Jiao and C. H. Chen. Customer requirement management in product development: A review of research issues. *Concurrent Engineering: Research and Applications*, 14(3): 173–185, 2006.
- Y. Jin and R. E. Levitt. The virtual design team: a computational model of project organizations. Working paper, Department of Civil Engineering, Stanford University, Stanford, 1996.
- Y. Jin, R. E. Levitt, and T. Christiansen. The virtual design team: a computer simulation framework for studying organizational aspects of concurrent design. Working paper, Center for Integrated Facility Engineering, Stanford University, Stanford, 1993.

- H. Kaindl. The missing link in requirements engineering. *ACM SIGSOFT Software Engineering Notes*, 2(18):30–39, 1993.
- K. C. Kapur and L. R. Lamberson. *Reliability in engineering design*. John Wiley and Sons, New York, 1st edition, 1977.
- J. Karlsson and K. Ryan. A cost-value approach for prioritizing requirements. *Journal of Software*, 14(5):67–74, 1997.
- R. Keller, C. Ecker, and J. P. Clarkson. Heuristics for change prediction. In *9th International Design Conference*, pages 873–880, Dubrovnik, Croatia, 2006.
- E. Koh and J. P. Clarkson. A modelling method to manage change propagation. In *17th International Conference on Engineering Design*, pages 253–264, Stanford, California, 2009.
- E. Koh, N. Caldwell, and P. Clarkson. A method to assess the effects of engineering change propagation. *Research in Engineering Design*, 23(4):329–351, 2012.
- A. Kossiakoff, W. Sweet, S. Seymour, and S. Biemer. *Systems engineering principles and practice*. Wiley, Hoboken, New Jersey, 2nd edition, 2011.
- G. Kotonya and I. Sommerville. *Requirements engineering: processes and techniques*. Wiley, Chichester, 1st edition, 1998.
- A. Kusiak and J. Wang. Dependency analysis in constraint negotiation. *Systems, Man and Cybernetics, IEEE Transactions on*, 25(9):1301–1313, 1995.
- A. Kuzel. *Sampling in Qualitative Inquiry*. Sage Publisher, 1999.
- W. Lam and V. Shankararaman. Requirements change: a dissection of management issues. In *Proceedings of the 25th EUROMICRO Conference*, Milan, Italy, September 1999.
- A. Lamsweerde. *Requirements engineering: from system goals to UML models to software specifications*. Wiley, 1st edition, 2009.

- N. Leibowitz, B. Baum, G. Enden, and A. Karniel. The exponential learning equation as a function of successful trials results in sigmoid performance. *Journal of Mathematical Psychology*, 54(3):338–340, 2010.
- Y. Lemmens, M. Guenov, A. Rutka, P. Coleman, and T. Schmidt-Schäffer. Methods to analyse the impact of changes in complex engineering systems. In *7th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference*, pages 1–15, Belfast, Ireland, 2007.
- V. Lévárdy. *Model-based framework for the adaptive development of engineering systems*. PhD thesis, Faculty of Mechanical Engineering, Technical University of Munich, Germany, 2006.
- V. Lévárdy and T. R. Browning. Adaptive test process designing a project plan that adapts to the state of a project. In *Proc of the 15th Annual International Symposium of INCOSE*, Rochester, USA, 2005.
- R. E. Levitt, J. Thomsen, T. R. Christiansen, J. C. Kunz, Y. Jin, and C. Nass. Simulating project work processes and organizations: toward a micro-contingency theory of organizational design. *Management Science*, 45(11):1479–1495, 1999.
- M. Lindvall. *An empirical study of requirements-driven impact analysis in object-oriented software evolution*. PhD thesis, Department of Computer and Information Science, Linköping, Sweden, 1997.
- S. L. Lohr. *Sampling: design and analysis*. Books Cole, Boston, MA, USA, 2nd edition, 2010.
- S. Lu, J. Cai, W. Burkett, and F. Udwadia. A methodology for collaborative design process and conflict analysis. *{CIRP} Annals - Manufacturing Technology*, 49(1):69–73, 2000.
- S.-Y. Lu, W. Elmaraghy, G. Schuh, and R. Wilhelm. A scientific foundation of collaborative engineering. *CIRP Annals - Manufacturing Technology*, 56(2):605 – 634, 2007.

- M. Lutz. *Learning Python*. O'Reilly, 3rd edition, 2009.
- C. M. Macal and M. J. North. Tutorial on agent-based modelling and simulation. *Journal of Simulation*, 4(3):151–162, 2010.
- C. M. Macal and M. J. North. Introductory tutorial: agent-based modeling and simulation. In *Proceedings of the 2011 Winter Simulation Conference (WSC)*, pages 1456–1469, 2011.
- C. Magee and O. de Weck. An attempt at complex system classification. In *ESD Internal Symposium*, number May, pages 1–34, 2002.
- M. Maher, J. Poon, and S. Boulanger. Formalising design exploration as co-evolution. In J. Gero and F. Sudweeks, editors, *Advances in Formal Design Methods for CAD*, IFIP The International Federation for Information Processing, pages 3–30. Springer US, 1996.
- J. Mattingly, W. Heiser, and D. Pratt. *Aircraft engine design*. American Institute of Aeronautics and Astronautics, Reston, Virginia, USA, 2nd edition, 2003.
- C. McAllister and T. W. Simpson. Multidisciplinary robust design optimization of an internal combustion engine. *Journal of Mechanical Design*, 125(1):124–130, 2003.
- C. McMahon and M. Xianyi. A network approach to parametric design integration. *Research in Engineering Design*, 8(1):14–31, 1996.
- H. McManus and D. Hastings. A framework for understanding uncertainty and its mitigation and exploitation in complex systems. In *MIT Engineering Systems Symposium*, March 2004.
- A. Messac and A. Ismail-Yahaya. Multiobjective robust design using physical programming. *Structural and Multidisciplinary Optimization*, 23(5):357–371, 2002. ISSN 1615-147X. URL <http://dx.doi.org/10.1007/s00158-002-0196-0>.



- S. Michael, G. Kilian, and B. Lucinne. Managing requirements or being managed by requirements-results of an empirical study. In *Proceedings of the 16th International Conference on Engineering Design (ICED07)*, Paris, France, August 2007.
- J. Mihm, C. Loch, and A. Huchzermeier. Problemsolving oscillations in complex engineering projects. *Management Science*, 49(6):733–750, 2003.
- H. Mills. The management of software engineering, part i: Principles of software engineering. *IBM Systems Journal*, 19(4):414–420, 1980.
- R. E. Moore. *Interval analysis*. Prentice-Hall, New Jersey, 1st edition, 1966.
- B. Morkos, P. Shankar, and J. D. Summers. Predicting requirement change propagation, using higher order design structure matrices: an industry case study. *Journal of Engineering Design*, 23(12):905–926, 2012.
- Z. P. Mourelatos and J. Zhou. Reliability estimation and design with insufficient data based on possibility theory. *AIAA Journal*, 43(8):1696–1705, 2005.
- R. Movahed-Khah, E. Ostrosi, and O. Garro. Analysis of interaction dynamics in collaborative and distributed design process. *Computers in Industry*, 61(1):2–14, 2010.
- T. Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, April 1989.
- NASA. *Systems engineering handbook*. National Aeronautics and Space Agency, 1st edition, 2007.
- NetLogo. Netlogo home page. <http://ccl.northwestern.edu/netlogo/>, 1999.
- W. L. Neuman. *Social research methods: qualitative and quantitative approaches*. Pearson International, sixth edition, 2006.
- E. Nikolaidis, S. Chen, H. Cudney, R. T. Haftka, and R. Rosca. Comparison of probability and possibility for design against catastrophic failure under uncertainty. *Journal of Mechanical Design*, 126(3):386–394, 2004.

- NIST. Integration definition for function modelling (idef0). Technical Report Publication 183, Federal Information Processing Standards (FIPS), National Institute of Standards and Technology, Gaithersburg, MD, USA, 1993.
- A. Nolan, S. Abrahao, P. Clements, and A. Pickard. Managing requirements uncertainty in engine control systems development. In *IEEE 19th International Requirements Engineering Conference*, pages 259–264, Trento, Italy, September 2011.
- B. ODonovan, C. Eckert, and P. Clarkson. Simulating design processes to assist design process planning. In *Proceedings of the ASME Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, pages 503–512, Salt Lake City, Utah, USA, 2004.
- J. Olsen, J. Cagan, and K. Kotovsky. Unlocking organisational potential: a computational platform for investigating structural interdependence in design. *Journal of Mechanical Design*, 131(3):031001–1, 2009.
- E. Ostrosi, L. Haxhiaj, and S. Fukuda. Fuzzy modelling of consensus during design conflict resolution. *Research in Engineering Design*, 23(1):53–70, 2012.
- K. Otto and K. Wood. *Product Design: Techniques in Reverse Engineering and New Product Development*. Prentice Hall, 1st edition, 2001.
- K. N. Otto and E. K. Antonsson. Design parameter selection in the presence of noise. *Research in Engineering Design*, 6(4):234–246, 1994.
- M. Ouertani. Supporting conflict management in collaborative design: An approach to assess engineering change impacts. *Computers in Industry*, 59(9):882–893, 2008.
- G. Pahl and W. Beitz. *Engineering design: A systematic approach*. Springer, 3rd edition, 2007.
- A. Pickard, A. Nolan, and R. Beasley. Certainty, risk and gambling in the development of complex systems. In *INCOSE International Symposium*, Chicago, USA, July 2010.

- P. Pikosz and J. Malmqvist. A comparative study of engineering change management in three swedish engineering companies. In *Proceedings of ASME design engineering technical conferences*, pages DETC98/EIM-5684, Atlanta, GA, USA, 1998.
- K. Pohl. *Requirements engineering: fundamentals, principles and techniques*. Springer-Verlag, Berlin Heidelberg, 1st edition, 2010.
- C. Poloni, P. Geremia, and A. Clarich. Multi-objective robust design optimization of an engine crankshaft. In C. Motasoaes, J. Martins, H. Rodrigues, J. Ambrsio, C. Pina, C. Motasoaes, E. Pereira, and J. Folgado, editors, *III European Conference on Computational Mechanics*, pages 386–386. Springer Netherlands, 2006. ISBN 978-1-4020-4994-1.
- A. Pritsker. Gert: graphical evaluation and review technique. Technical Report RM-4973-NASA, The RAND Corporation, Pittsburgh, Pennsylvania, April 1966.
- A. Rao and M. Georgeff. Modeling rational agents within a bdi-architecture. In *Second International Conference on Principles of Knowledge Representation and Reasoning*, San Mateo, C.A., 1991.
- A. Rao and M. Georgeff. Bdi agents: from theory to practice. In *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95)*, San Francsisco, USA, June 1995.
- E. Rechtin. *Systems Architecting*. Prentice-Hall, New York NY, USA, 1st edition, 1991.
- Repast. Repast home page. <http://repast.sourceforge.net/>, 2006.
- S. Robertson and J. Robertson. *Mastering the Requirements Process*. Addison-Wesley, Amsterdam, Netherlands, 2nd edition, 2006.
- C. Robson. *Real World Research - A Resource for Social Scientists and Practitioners*. Blackwell Publishing, 2nd edition, 2002.
- Rolls-Royce. *The Jet Engine*. Key Publishing, 5th edition, 2011.

Rolls-Royce. <http://www.rolls-royce.com/>, 2013.

J. Rumbaugh, I. Jacobson, and G. Booch. *The unified modeling language reference manual*. Addison-Wesley, Reading, Massachusetts, US, 1999.

S. Russel and P. Norvig. *Artificial intelligence: a modern approach*. Prentice-Hall, Englewood Cliffs, N.J., 1995.

A. Rutka, M. Guenov, and Y. Lemmens. Methods for engineering change propagation analysis. In *25th International Congress of the Aeronautical Sciencesl Sciences*, pages 1–8, Hamburg, Germany, 2006.

T. Saaty. *The analytic hierarchy process*. RWS Publications, 1st edition, 1990.

A. Sage and J. Armstrong. *Introduction to systems engineering*. Wiley, 1st edition, 2000.

H. Saravanamutoo, G. Rogers, and H. Cohen. *Gas turbine theory*. Pearson Education Ltd., Dorling Kindersley, India, 5th edition, 2001.

G. Shaffer. *A mathematical theory of evidence*. Princeton University Press, Princeton, NJ, 1st edition, 1976.

O. Shai and Y. Reich. Infused design. i. theory. *Research in Engineering Design*, 15(2): 93–107, 2004.

M. Shaw and B. Gaines. Requirements acquisition. *Software Engineering Journal*, 11(3): 149–165, 1996.

A. Sinha, N. Bera, J. K. Allen, J. H. Panchal, and F. Mistree. Uncertainty management in the design of multiscale systems. *Journal of Mechanical Design*, 135(1):011008, 2013.

I. Sommerville. *Software engineering: international version*. Prentice Hall, 9th edition, 2010.

Standish. The chaos report. Technical Report MSU-CSE-99-39, The Standish Group International, Boston, December 1995.

- J. D. Sterman. System dynamics: systems thinking and modeling for a complex world. Working Paper ESD-WP-2003-01.13-ESD Internal Symposium, Massachusetts Institute of Technology, Engineering Systems Division, Boston, May 2002.
- D. V. Steward. Partitioning and tearing systems of equations. *SIAM Numerical Analysis*, 2(2):345–365, 1965.
- M. N. Sudin and S. Ahmed. Investigation of change in specifications during a product’s lifecycle. In *Proceedings of the 17th International Conference on Engineering Design (ICED09)*, pages 371–380, Paolo Alto, CA, USA, August 2009.
- M. N. Sudin and S. Ahmed-Kristensen. Change in requirements during the design process. In *Proceedings of the 18th International Conference on Engineering Design (ICED11)*, pages 200–208, Copenhagen, Denmark, August 2011.
- N. P. Suh. *The principles of design*. Oxford University Press, 1990.
- S. Sundaresan, K. Ishii, and D. R. Houser. A robust optimization procedure with variations on design variables and constraints. *Engineering Optimization*, 24(2):101–117, 1995.
- H. Takeda, A. Tsumaya, and T. Tomiyama. Synthesis thought processes in design. In *CIRP International Design Seminar*, page 249268, University of Twente, Enschede, the Netherlands, 1999.
- D. P. Thunnissen. Uncertainty classification for the design and development of complex systems. In *Proceedings of the 3rd Annual Predictive Methods Conference, Veros Software*, 2003.
- D. P. Thunnissen. *Propagating and mitigating uncertainty in the design of complex multidisciplinary systems*. PhD thesis, California Institute of Technology, USA, 2005.
- M. M. Tseng and J. Jiao. Computer-aided requirement management for product definition: a methodology and implementation. *Concurrent Engineering: Research and Applications*, 6(3):145–160, 1999.

- K. L. Tsui. An overview of taguchi method and newly developed statistical methods for robust design. *IIE Trans.*, 24(5):44–57, 1992.
- K. T. Ulrich and S. D. Eppinger. *Product Design and Development*. McGraw-Hill Higher Education, Boston MA, USA, fourth edition, 2008.
- US DoD Systems Management College. *Systems Engineering Fundamentals*. Defense Acquisition University Press, 2001.
- H. Veeke, G. Lodewijks, and J. Ottjes. Conceptual design of industrial systems: an approach to support collaboration. *Research in Engineering Design*, 17(2):85–101, 2006.
- G. Vianello and S. Ahmed-Kristensen. A comparative study of changes across the lifecycle of complex products in a variant and a customised industry. *Journal of Engineering Design*, 23(2):99–117, 2012.
- P. Walley. *Statistical reasoning with imprecise probabilities*. Chapman Hall, London, UK, 1st edition, 1991.
- J. Wardekker, J. van der Sluijs, P. Janssen, P. Klopprogge, and A. Petersen. Uncertainty communication in environmental assessments: views from the dutch science-policy interface. *Environmental Science & Policy*, 11(7):627–641, 2008.
- M. Weber and J. Weisbrod. Requirements engineering in automotive development: experiences and challenges. *IEEE Software*, 20(1):1624, 2003.
- D. Whitney. Manufacturing by design. *Harvard Business Review*, (July), 1988.
- K. E. Wiegers. *Software Requirements*. Microsoft Press, 2nd edition, 2010.
- J. D. Wiest. *A management guide to PERT/CPM*. Prentice-Hall, 1969.
- K. L. Wood, E. K. Antonsson, and J. L. Beck. Representing imprecision in engineering design: comparing fuzzy and probability calculus. *Research in Engineering Design*, 1(3-4):187–203, 1990.

- M. Wooldridge. *An introduction to multiagent systems*. John Wiley & Sons, 2nd edition, 2009.
- M. Wooldridge and N. Jennings. Intelligent agents: theory and practice. *Knowledge engineering review*, 10(2):115–162, 1995.
- D. Wynn. *Model-based approaches to support process improvement in complex product development*. PhD thesis, University of Cambridge, Cambridge, UK, 2007.
- D. Wynn, C. Eckert, and P. Clarkson. Applied signposting: a modelling framework to support design process improvement. In *Proceedings of the ASME International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, pages DETC2006–99402, Philadelphia, Pennsylvania, USA, 2006.
- D. Wynn, C. Eckert, and P. Clarkson. Modelling iteration in engineering design. In *Proceedings of the 16th International Conference on Engineering Design*, pages 693–694, Paris, France, 2007.
- D. C. Wynn, K. Grebici, and P. Clarkson. Modelling the evolution of uncertainty levels during design. *International Journal on Interactive Design and Manufacturing (IJI-DeM)*, 5(3):187–202, 2011.
- A. Xiao, S. Zeng, J. Allen, D. Rosen, and F. Mistree. Collaborative multidisciplinary decision making using game theory and design capability indices. *Research in Engineering Design*, 16(1-2):57–72, 2005.
- L. E. Yelle. The learning cuve: historical review and comprehensive survey. *Decision Sciences*, 10(2):302–328, 1979. ISSN 1540-5915.
- R. K. Yin. *Case study research design and methods*. SAGE PUBLICATIONS, 1984.
- L. A. Zadeh. Fuzzy sets. *Information Control*, 8:338–353, 1965.
- Z. Zhang. Effective requirements development - a comparison of requirements elicitation techniques. In E. Berki, J. Nummenmaa, I. Sunley, M. Ross, and G. Staples, editors,

*Software Quality Management XV: Software Quality in the Knowledge Society*, pages 225–240. British Computer Society, 2007.

H. J. Zimmermann. An application-oriented view of modeling uncertainty. *European Journal of Operational Research*, 122(2):190–198, 2000.



# Appendix A

## Exploratory research

This appendix collects the interview guide used during exploratory research performed by the author at Rolls-Royce. As described during Chapter 1, a series of 30 open questions were prepared and conducted during the course of semi-structured interviews to a purposive sample of 14 engineers and managers. The interview guide consisted of the following questions:

### Introduction

- Interviewer presentation, personal background and description of the PhD topic
- Explanation of the purpose of the interview, defining the scope of the research and the main problems to be tackled by research
- Ensuring confidentiality and asking for permission for notes
- Request for presentation from the interviewee: current position, number of years of professional experience, years of experience in product development, years of experience at Rolls-Royce, previous positions at R-R, etc

### Warm-up

- Overview of the requirements process and database in the organization
  - What are the main phases in the requirements process?
  - What is the role of the different corporate units involved?

- How are higher level requirements cascaded to lower levels?
- How are decisions taken?
- Who writes requirements?
- What is the landscape of requirements documents written?
- How is the requirements database organized and managed?

## **Main body**

- Behavior and beliefs relatively to requirements uncertainty and change
  - Is requirements change considered normal?
  - How do people deal with it when change occurs?
  - How do engineers deal with uncertainty/risk in requirements?
- Response to uncertainty and risks
  - If requirements are uncertain, how is the uncertainty or the maturity level in requirements identified and assessed?
  - How is uncertainty cascaded from higher level requirements to lower levels?
  - How are risks in requirements managed?
- Knowledge about the causes of change
  - What are the typical causes of requirements change? Why?
  - Could you provide some examples?
  - Which are internal to the organization? Which ones are external?
  - How is change communicated?
  - How are the changes and their causes documented?
- Knowledge about the impact of change
  - When changed, which requirements hurt the most? Which ones can propagate across the system?
  - What is the impact?
  - Could you provide some examples?

- How much scrap and rework is there from requirements change?
- Is the impact of changes in requirements usually anticipated? How?
- What sort of late changes in requirements have been found?

## Cool-Off

- Problems and challenges in requirements change
  - What are the main problems or challenges to be tackled by research on this topic? What is your personal view on this?
  - According to your experience, which projects/work packages/ programs would be interesting case-studies? Why?
  - What characteristics should projects have to qualify for requirements research?
- Results from previous research at Rolls-Royce
  - Which research projects related with Requirements have already been performed at the company? Who was involved?
  - What was the methodology followed?
  - What results have been found?

## Closure

- Acknowledging the interviewee for his time
- Request for related contacts that may be of interest for this research
- Hand-on the door for additional questions and information

In addition, this Appendix presents also specific information about the sample of engineers and managers that have been interviewed during the course of the exploratory research performed by the author. The name, job titles and main area of expertise of each interviewee is shown in Table A.1.

Table A.1: The sample according to the interviewee's type or work or area of expertise.

Name	Current Job Title	Type of Work / Area of Expertise
Neil Armstrong	Design Specialist	Design Methods
Andy Nolan	Chief of Software Improvement	Engineering Improvement and Quality
Gareth Doyle	Programme Manager in Strategic Programmes	Strategic Management
Alistair Marvin	Requirements Specialist	Requirements Engineering
Lee Glazier	Requirements Specialist and Chief of Word Class Systems	Requirements Engineering
Keith Howells	Chief of Integrated Design and Manufacture Management	Systems Engineering
Andy Pickard	Chief of Improvements and Quality, Control Systems	Engineering Improvement and Quality
Richard Simkins	Integrated Project Team Leader/ Sub-system integrator	Sub-System Integration
Mike Moss	Head of Systems Design Integration Team	Design Methods
Rez Manzori	Sub-System integrator	Sub-System Integration
Martyn Richards	Future Programs Engineering Design Specialist	Concept Development & Selection
Richard Beasley	Chief of Systems Engineering	Systems Engineering
Gareth Armstrong	Design Technologist	Design Methods
Dave Yazdani	Head of Systems Engineering	Detailed Design & Development

# Appendix B

## Empirical study of root causes

This appendix presents additional details of the knowledge elicitation sessions that took place in the empirical study to the root causes of requirements change performed by the author.

The Rolls-Royce engineers which were in the role of requirements managers during the development project investigated in the study were Kyle Martin, Carol McCorkindale, Heather Dunbar and John Bentley. These requirements management were invited and accepted to participate in the case-study designed by the author.

Figure B.1 presents the template prepared by the author in Microsoft Excel and distributed to these engineers during the knowledge elicitation sessions described in Chapter 3. It illustrates that the template contained the new version of the requirement, the previous version, the type of change and the list of causes of change, which was available from a drop down menu.

At the begging of the session, the author explained also to the engineers the guidelines for filling the template during the root cause assessment of each requirement change in the sample defined for the study. The guidelines can be also depicted in Figure B.1. The engineers were asked to read the version of the requirements observed at the issues  $N$  and  $N - 1$  of the PRD and the type of change observed. They were then asked to select the primary reason of change from the drop-down menu. If a secondary reason of change was found, they were invited to repeat the previous task. Finally, a space was reserved to register additional explanations from the expert (Figure B.1).

Read ID	Read requirement released at version N-1 and version N of PRD read type of change		and	Select primary reason for change from Drop-Down Menu	Select secondary reason for change, if it exists	Provide comments if needed
ID	Requirement at PRD4	Requirement at PRD5	Type of change	The requirement changed because...	And also because... (Optional)	Additional Comments (Optional)
1 XXX:PRD-6	The highest rated engine BOM shall be capable of down rating to operation at all certified engine rating structures via substitution of the Data Entry plug as the only hardware change. [XXX:BRD3, XXX:BRD6]		Deleted			
2 XXX:PRD-2197		The Engine Mounts shall be designed in order to allow Aircraft certification under the applicable FAR25 regulations. {TRACE: XXX:BRD-32}	New			
3 XXX:PRD-1960		Accessories (LRUs) should be common between the XXX-A and XXX-C Engines. {TRACE: XXX:BRD-3465}	New			
4 XXX:PRD-1911		The Engine shall be designed to maximise customer value on the YYY-A aircraft with a technology insertion plan to achieve YYY-B thrust requirements. {TRACE: XXX:BRD-3}	New	<div> The customer changed the requirement  Regulation or certification changed  Business requirements changed  The requirement was missing  The requirement was unachievable  The requirement was unverifiable  The requirement was redundant  The requirement was decomposed </div>		
5 XXX:PRD-9	All components shall be designed for certification at XXXIbf AET rating, (e.g. pressure, containment, overspeed), with the exception of those detailed in XXX:PRD005. [XXX:BRD3]		Deleted			
6 XXX:PRD-8	The Engine shall be capable of entry into service at XXXIbf 2 years after the initial XXX-A product EIS, as defined in the XXX Master Programme. [XXX:BRD3, XXX:BRD5]		Deleted			
XXX:PRD-1558		The Engine shall meet the requirements of this document within the Aircraft flight envelope defined in Figure 4. {TRACE: XXX:Ds:R-8 [EDs-4187],	New			

Figure B.1: Template prepared by the author and distributed during the knowledge elicitation sessions with the requirements managers. Contains disguised information due to confidentiality.

# Appendix C

## Agent model of early design

This appendix presents additional details about the main classes and methods which have been developed for the Agent Model for Planning and rEsearch of eaRly dEsign (AMPERE), thus complementing the description which has been provided in Chapter 6 and supporting specifically the reader’s understanding of Section 6.3.3. Description of the main internal classes, methods and attributes contained in the Design Agent, Task and Solution classes are provided in Tables C.1, C.2 and C.3.

Table C.1: Description of main internal elements contained in the Design Agent class.

Type	Name	Description
Class	Reasoning	A class containing a Belief-Desire-Intent (BDI) reasoning engine for deliberation and planning of design activities. Implemented as a subclass of the SPADE Behaviour class. Cyclic behaviour calling in a sequential manner the methods <i>Brf()</i> , <i>Options()</i> , <i>Filter()</i> , <i>Plan()</i> and <i>Execute()</i> .

Continued on next page

Table C.1: Continued.

Type	Name	Description
	Time Update	A class for continuous update and synchronization of the current time during simulation. Implemented as a subclass of the SPADE Event class.
Methods	Brf()	Revises the agent's Beliefs database. Declared as an abstract method, i.e., must be implemented by child agents.
	Options()	Returns a list of possible Desires based on the current status of the agent's Beliefs. Declared as an abstract method, i.e., must be implemented by child agents.
	Filter()	Filters the current Desires according to the agent's Beliefs and returns one Intention. Declared as an abstract method, i.e., must be implemented by child agents.
	Plan()	Plans what is the task the agent will perform according to his internal library of available tasks. Loops through each element of the library and picks the element(s) with a postcondition similar to the agent's intention and with a precondition that holds true in a logical test.

Continued on next page



Table C.1: Continued.

Type	Name	Description
	Execute()	Loops through the tasks that have been selected and incorporated in the agent's plan during the planning step and executes them in a sequential manner.
	setup()	An abstract method which must be implemented in SPADE child agents. Sets-up the agent's Belief-Desire-Intention data structures, the agent's attributes at initialization and adds the Reasoning and Time Update behaviours, which are loaded when the agent is instanced.
Attributes	<ul style="list-style-type: none"> <li>• Name</li> <li>• Experience</li> <li>• Beliefs</li> <li>• Desires</li> <li>• Intentions</li> <li>• Plan</li> <li>• Task Library</li> <li>• Clock</li> </ul>	The Name attribute stores the agent's identification in the SPADE platform and experience refers to the level of design experience accumulated by the agent (Junior, Senior, etc). Beliefs, Desires, Intentions, Plan and Task Library are attributes storing data. Beliefs have been implemented through a dictionary structure, which allows to associate a belief entry to a string, float, logical condition, list or other object defining its content. The remaining are implemented using lists of objects. The Clock attribute keeps track of the current time.

Table C.2: Description of main internal elements contained in the Task class.

Type	Name	Description
Methods	Task()	Constructor method, sets-up major attributes of the object. Construction requires passing values for Name, Precondition, Postcondition, Standard Duration and Initial Duration and a pointer to the agent holding the task object.
	Body()	A method with task's body, i.e., the individual statements to be performed by the agent when executing the task. Declared as an abstract method, i.e., must be implemented by child tasks.
	GenInitDur()	Called at object construction to generate the initial task time from probability density function defined according to the agent's level of experience.
	GenStanDur()	Generates the standard duration of tasks from probability density function defined according to the agent's level of experience. Generates variations in the task's standard duration to account for day-to-day execution variability.

Continued on next page

Table C.2: Continued.

Type	Name	Description
	GenImprovRate()	Generates the task's progress rate from probability density function defined according to the agent's level of experience. Generates variations in the progress rate to account for day-to-day execution variability.
	GetDuration()	Computes and updates the task duration attribute. Calls the <i>GenStanDuration()</i> and <i>GenImprovementRate</i> to generate variability.
	SwitchMode()	Shifts the task's mode attribute from individual to collaboration and vice-versa. Called when agents engage in collaboration and has impact over the task's progress rate.
Attributes	<ul style="list-style-type: none"> <li>• Name</li> <li>• Precondition</li> <li>• Postcondition</li> <li>• Duration</li> <li>• Mode</li> <li>• NIter</li> </ul>	<p>The Name attribute stores the task's identification. Precondition holds a logical statement than needs to be true according to agent's belief status so that the task may be executed. Postcondition contains a description of what the task performs when executed and that may match the agent's intentions during planning. Duration stores the time that the task will require when it is executed. Mode describes whether the task is being performed in collaboration or not. The NIter attribute keeps track of the number of task iterations performed by the agent.</p>

Table C.3: Description of main internal elements contained in the Solution class.

Type	Name	Description
Methods	Solution()	Constructor method, sets-up major attributes of the object. Construction requires passing values for Name and a pointer to the agent holding the solution object.
	GenInitQuality()	Called at object construction to generate generate initial quality level from probability density function defined according to the agent's level of experience.
	GenImprovRate()	Generates the solutions's progress rate from probability density function defined according to the agent's level of experience. Generates variations in the progress rate to account for day-to-day performance variability.
	GetQuality()	Computes and updates the solution quality attribute. Calls the <i>GenImprovementRate</i> to generate variability and the <i>UpdateIter()</i> to keep track of the number of iterations performed.
	UpdateIter()	Updates the number of iterations performed by agent. Called after activity execution leading to a new or improved design solution.

Continued on next page

Table C.3: Continued.

Type	Name	Description
	KnockDown()	Deteriorates the solution quality from a probability density function and proportionally to the magnitude of change that has been detected by the agent. Called after the arrival of a change event.
	SwitchMode()	Shifts the solution's mode attribute from individual to collaboration and vice-versa. Called when agents engage in collaboration and has impact over the solution's progress rate.
Attributes	<ul style="list-style-type: none"> <li>• Name</li> <li>• Quality</li> <li>• Mode</li> <li>• NIter</li> </ul>	<p>The Name attribute stores the design solution's identification. Quality stores the value of optimization achieved through the agent's design activities. Mode describes whether the design solution is being worked in collaboration or not. The NIter attribute keeps track of the number of solution generation iterations performed by the agent.</p>