



TECHNISCHE UNIVERSITÄT  
CHEMNITZ



**BOSCH**

# Development of New Model-based Methods in ASIC Requirements Engineering

## Master Thesis

Submitted in Partial Fulfilment of  
the Requirements for the Academic Degree

### Master of Science (M.Sc.)

Department of Computer Science  
Chair of Computer Engineering

Submitted by: Chukwuma Onuoha Onuoha

Matriculation Number: 458304

Date: 30.07.2020

Supervising tutors: Prof. Dr. Dr. h. c. Wolfram Hardt  
Dipl.-Inf. Michael Nagler

Resident supervisor: M.Sc. Aljoscha Kirchner

*This page intentionally left blank*

# Acknowledgements

My entire academic journey, particularly this thesis, would be extremely difficult, if not impossible, without the assistance and support of those who directly contributed towards this journey in very many different, but important ways, including:

Mom and dad, Chukwudi, Ogechi, Ikechukwu, Onyinyechi, Uchechi and Toochukwu, and families: eternal gratitude to you, dearest parents and siblings!

Jude Nwokey and family: “ekele kelere m unu!”; Nkiruka and Eugene Aniekwe, and family: “speechless!”, is how I feel; Bibiane and Armel, Gisela and Peter Tews: “ganz lieben Dank!”; Tina and Martin, Josy and Martin, Elli, Kathrin: “für immer und ewig!”. Abdelbar, Aliu, Coco, Happiness, Ritwik and Zeeshan: thank you guys!

This work has been developed in the ITEA3 project COMPACT (reference number: 16018). COMPACT is funded by the Austrian Research Promotion Agency FFG (project number 863828 and 864769), the Finnish funding agency for innovation Tekes (Diary Number 3098/31/1717) and the German ministry of education and research (BMBF) (reference number: 01IS17028). The authors are responsible for the content of this publication.

The entire AE/NE-IC staff of the Robert Bosch GmbH, Reutlingen, most especially to Dr. Jan-Hendrik Oetjens, Dr. Andreas Mauderer and Anja Hammermann: the support you provided will be dearly and eternally cherished; Horst Berger, Mores Assogba, Markus Friedrich, Olaf Kleinwegen and Torsten Mahl: you will always have a special place in my heart; to Stephan Sachs: I will remember to water the flowers everywhere I go; to mention just a few. From the bottom of my heart: thanks to you all!

The guidance and support of Dipl.-Inf. Michael Nagler is a joy to behold! Those listening ears were a soothing relief. Each time. Every time. When the goings were tough, you were always there to help keep me strong and going. Thank you.

I could not have asked for a better supervisor than Aljoscha Kirchner, nor does a better support exist anywhere else. Thank you for giving me this opportunity, trusting me to make decisions, supporting every decision, then believing in me through thick and thin. By associating with you, I have learnt to be a mentor.

Few people in life will have such gracious and positive influence on so many as that which always brings students to the doorstep of Prof. Dr. Wolfram Hardt. Thank you for guiding me through my studies, and especially for supervising this thesis.

And, to God Almighty who gives me the sole-inspiration to carry on I say, Father a glorious thanks to You!

# Abstract

Requirements in the development of application-specific integrated circuits (ASICs) continue to increase. This leads to more complexities in handling and processing the requirements, which often causes inconsistencies in the requirements. To better manage the resulting complexities, ASIC development is evolving into a model-based process. This thesis is part of a continuing research into the application and evolution of a model-based process for ASIC development at the Robert Bosch GmbH. It focuses on providing methodologies that enable tracing of ASIC requirements and specifications as part of a model-based development process to eliminate inconsistencies in the requirements. The question of what requirements are and, what their traceability means, is defined and analysed in the context of their relationships to models.

This thesis applies requirements engineering (RE) practices to the processing of ASIC requirements in a development environment. This environment is defined by availability of tools which are compliant with some standards and technologies. Relying on semi-formal interviews to understand the process in this environment and what stakeholders expect, this thesis applies the standards and technologies with which these tools are compliant to provide methodologies that ensures requirements traceability.

Effective traceability methods were proven to be matrices and tables, but for cases of fewer requirements (ten or below), requirement diagrams are also efficient and effective. Furthermore, the development process as a collaborative effort was shown to be enhanced by using the resulting tool-chain, when the defined methodologies are properly followed. This solution was tested on an ASIC concept development project as a case study.

**Keywords:** Artifact, Model, Requirement, Specification, Traceability

# Contents

	Page
<b>Acknowledgements</b> . . . . .	<b>iii</b>
<b>Abstract</b> . . . . .	<b>iv</b>
<b>List of Figures</b> . . . . .	<b>viii</b>
<b>List of Tables</b> . . . . .	<b>x</b>
<b>List of Abbreviations</b> . . . . .	<b>xi</b>
<b>1 Introduction</b> . . . . .	<b>1</b>
1.1 Motivation . . . . .	2
1.1.1 Aim and scope . . . . .	4
1.1.2 Research objectives . . . . .	5
1.2 Research approach . . . . .	6
1.2.1 Action design research . . . . .	6
1.2.2 Questionnaire-based gap analysis . . . . .	6
1.3 Professorship of Computer Engineering at the Technische Universität Chemnitz . . . . .	8
1.4 Partner organisations . . . . .	9
1.5 Structure of this thesis . . . . .	9
1.6 Summary . . . . .	10
<b>2 Technical background</b> . . . . .	<b>11</b>
2.1 Application-specific integrated circuit . . . . .	11
2.2 Requirements engineering process and management . . . . .	13
2.2.1 Requirements elicitation . . . . .	17
2.2.2 Requirements analysis . . . . .	18
2.2.3 Requirements specification and documentation . . . . .	18
2.2.4 Requirements verification and validation . . . . .	19
2.3 Systems engineering . . . . .	19
2.3.1 Natural languages in systems engineering . . . . .	20
2.3.2 Document-based approach to systems engineering . . . . .	21
2.3.3 Model-based approach to systems engineering . . . . .	21
2.4 Unified Modelling Language . . . . .	22

2.5	Systems Modelling Language . . . . .	23
2.5.1	Behavioural diagrams . . . . .	26
2.5.2	Structural diagrams . . . . .	28
2.6	Summary . . . . .	30
<b>3</b>	<b>State of the art . . . . .</b>	<b>31</b>
3.1	Origin of this thesis . . . . .	31
3.2	Reliance on natural languages . . . . .	33
3.3	Document-based systems engineering . . . . .	34
3.4	Model-based systems engineering . . . . .	37
3.5	Application lifecycle management . . . . .	39
3.6	Change and configuration management (CCM) . . . . .	40
3.6.1	Change management . . . . .	40
3.6.2	Configuration management . . . . .	41
3.6.3	Source control management and version control . . . . .	41
3.7	Quality management (QM) . . . . .	42
3.8	Open services for lifecycle collaboration . . . . .	42
3.9	Traceability methodologies and types . . . . .	47
3.10	Summary . . . . .	51
<b>4</b>	<b>Concepts . . . . .</b>	<b>52</b>
<b>5</b>	<b>Implementation system . . . . .</b>	<b>55</b>
5.1	IBM <sup>®</sup> Rational <sup>®</sup> . . . . .	55
5.1.1	DOORS <sup>®</sup> Next Generation . . . . .	56
5.1.2	Rhapsody <sup>®</sup> (RR) . . . . .	57
5.1.3	Team Concert <sup>®</sup> (RTC) . . . . .	58
5.1.4	Rhapsody <sup>®</sup> Model Manager (RMM) . . . . .	58
5.1.5	Jazz <sup>™</sup> Team Server . . . . .	59
5.2	Traceability in RR . . . . .	60
5.2.1	Requirements tables . . . . .	63
5.2.2	Matrix views . . . . .	64
5.2.3	Annotations, relations and tags in RR . . . . .	64
5.3	Summary . . . . .	65
<b>6</b>	<b>Integration and deployment . . . . .</b>	<b>66</b>
6.1	Linking the project areas (PAs) . . . . .	68
6.2	Adding CCM and QM to the lifecycle PA . . . . .	72
6.3	The case for a naming convention . . . . .	73
6.4	Application of attributes . . . . .	74
6.5	Handling RMM-based models . . . . .	76
6.5.1	Delivery to RMM . . . . .	77
6.5.2	Collaboration and harmonious modelling . . . . .	77
6.5.3	Extending views and tags . . . . .	79

6.6	Summary . . . . .	82
<b>7</b>	<b>Case study-based evaluation . . . . .</b>	<b>83</b>
7.1	Consistency of requirements . . . . .	84
7.2	A working tool-chain . . . . .	90
7.3	Evaluation of matrix views and tables . . . . .	93
7.4	Recommendations . . . . .	97
7.5	Summary . . . . .	99
<b>8</b>	<b>Conclusion . . . . .</b>	<b>100</b>
8.1	Summary . . . . .	100
8.2	Future research . . . . .	102
	<b>References . . . . .</b>	<b>XIV</b>

# List of Figures

1.1	Research methodology . . . . .	7
2.1	Example of on-chip Architecture of smart-sensor ASIC . . . . .	12
2.2	ASIC development workflow . . . . .	13
2.3	RE as a process . . . . .	14
2.4	Workflow in the RE process . . . . .	15
2.5	Research methodology . . . . .	23
2.6	Sample of a SysML-based model . . . . .	24
2.7	SysML hierarchy . . . . .	26
2.8	SysML behavioural diagram . . . . .	27
2.9	SysML structural diagram . . . . .	28
3.1	Workflow for a model-based ASIC development . . . . .	32
3.2	Classic requirement miscommunication example in system development	34
3.3	Requirements documentation name mismatch . . . . .	35
3.4	Verification and validation with DBSE . . . . .	36
3.5	Exhibition of the use of OSLC in a server-client environment . . . . .	43
3.6	Integration of systems using OSLC . . . . .	46
3.7	Associating models to requirements artifacts . . . . .	50
3.8	Associating requirements to other related documents . . . . .	51
5.1	Comparing the context views of DOORS and DNG . . . . .	56
5.2	Illustration of components and streams . . . . .	59
5.3	Sample view of a RR Requirement Diagram based . . . . .	61
5.4	Sample of unreadable requirement diagram . . . . .	62
5.5	Comparison of context patterns in table layout . . . . .	63
6.1	Structure of ALM . . . . .	66
6.2	Structure of JTS-based platform for integration . . . . .	67
6.3	Sample structure of multiple PAs . . . . .	68
6.4	Project setup for DNG and RMM . . . . .	69
6.5	Communication path between local and remote repositories . . . . .	70
6.6	RTC Client-workspace connection on a workstation . . . . .	70
6.7	Comparing the context views of DOORS and DNG . . . . .	71
6.8	Change sets as a function of CCM . . . . .	72
6.9	Creating the required artifact types . . . . .	74
6.10	The central role of change sets . . . . .	77



6.11	Behaviour of single RMM PA and RTC Client with multiple-workspaces .	78
6.12	Change set notification pattern . . . . .	79
6.13	Setting the scope for table views . . . . .	80
6.14	Default table view . . . . .	80
6.15	Behaviour of single RMM PA and RTC Client with multiple-workspaces .	82
7.1	Default table view . . . . .	83
7.2	Method at the start of the research . . . . .	84
7.3	DNG-based requirements available in RR model . . . . .	85
7.4	Sample DNG view of FTR_Concept_Project's requirements . . . . .	86
7.5	DNG-based requirements used in a block diagram . . . . .	86
7.6	Remote requirements as internet links . . . . .	88
7.7	Requirement diagrams with remote artifacts . . . . .	89
7.8	Display of remote artifacts linked to RR models in RR . . . . .	91
7.9	Access management using multiple components . . . . .	92
7.10	Confirmation of remote DNG artifacts in a matrix view . . . . .	94
7.11	Display of available requirement artifacts . . . . .	95
7.12	Notification for obsolete requirement artifacts . . . . .	96
7.13	Decision process for using the platform . . . . .	98

# List of Tables

6.1	Sample of data types in DNG . . . . .	75
6.2	Sample of attributes in DNG . . . . .	75
6.3	Sample of DNG artifact types . . . . .	76

# List of Abbreviations

<b>ADD</b>	Architecture Description Document
<b>ADR</b>	Action Design Research
<b>AE</b>	Automotive Electronics
<b>ALM</b>	Application Lifecycle Management
<b>AM</b>	Architecture ManagementAM
<b>API</b>	Application Program(ming) Interface
<b>ASE</b>	Automotive Software Engineering
<b>ASIC</b>	Application-Specific Integrated Circuit
<b>AUTOSAR</b>	AUTomotive Open System Architecture
<b>CCM</b>	Change and Configuration Management
<b>CD</b>	Continuous Delivery, Continuous Deployment
<b>CENELEC</b>	European Committee for Electrotechnical Standardizatio
<b>CI</b>	Continuous Integration
<b>CLM</b>	Rational <sup>®</sup> Collaboration for Lifecycle Management
<b>CR</b>	Customer Requirements
<b>CRUD</b>	Create, Read, Update, Delete
<b>CT</b>	Continuous Testing
<b>DBSE</b>	Document-Based Systems Engineering
<b>DevOps</b>	Development and Operations
<b>DNG</b>	IBM <sup>®</sup> Rational <sup>®</sup> DOORS <sup>®</sup> Next Generation
<b>DO</b>	Document in aeronautic systems
<b>DOORS</b>	IBM <sup>®</sup> Rational <sup>®</sup> Dynamic Object-Oriented Requirements <sup>®</sup>
<b>EAST-ADL</b>	Electronic Architecture STudy - Architecture Description Language

<b>ECU</b>	Electronic Control Unit
<b>EN</b>	European Standards
<b>FPGA</b>	Fiel-Programmable Gate Array
<b>FS</b>	Functional Specification
<b>FSV</b>	Formal Specification and Verification
<b>Hw</b>	Hardware
<b>IBM</b>	International Business Machines <sup>®</sup>
<b>ID</b>	Identification number
<b>IEC</b>	International Electrotechnical Commission
<b>INCOSE</b>	International Council on Systems Engineering
<b>ISO</b>	International Organization for Standardisation
<b>IT</b>	Information Technology
<b>JSON</b>	JavaScript Object Notation
<b>JTS</b>	Jazz <sup>™</sup> Team Server
<b>LHS</b>	Left Hand Side
<b>MARTE</b>	Modelling and Analysis of Real Time Embedded systems
<b>MBSE</b>	Model-Based Systems Engineering
<b>NL</b>	Natural Language
<b>NLP</b>	Natural Language Processing
<b>OEM</b>	Original Equipment Manufacturer
<b>OMG</b>	Object Management Group <sup>®</sup>
<b>OSLC</b>	Open Services for Lifecycle Collaboration
<b>PA</b>	Project Area
<b>Pkg</b>	Package (in SysML)
<b>PLM</b>	Product Lifecycle Management
<b>QBGA</b>	Questionnaire-Based Gap Analysis
<b>QM</b>	Quality Management

<b>RDF</b>	Resource Description Framework
<b>RE</b>	Requirements Engineering
<b>REST</b>	REpresentational State transfer
<b>RHS</b>	Right Hand Side
<b>RM</b>	requirements Management
<b>RMM</b>	IBM <sup>®</sup> Rational <sup>®</sup> Rhapsody <sup>®</sup> Model Manager
<b>RO</b>	Read Only
<b>RR</b>	IBM <sup>®</sup> Rational <sup>®</sup> Rhapsody <sup>®</sup>
<b>RTC</b>	IBM <sup>®</sup> Rational <sup>®</sup> Team Concert <sup>®</sup>
<b>RTVM</b>	Requirements Traceability Verification Matrices
<b>SCM</b>	Source Configuration Management
<b>SDLC</b>	System Development Life Cycle
<b>SE</b>	Systems Engineering
<b>SiP</b>	Systems-in-Package
<b>SoC</b>	Systems-on-a-Chip
<b>SPI</b>	Serial Peripheral Interface
<b>Sw</b>	Software
<b>SysML</b>	Systems Modelling Language
<b>TTM</b>	Time-to-Market
<b>TUC</b>	Technische Universität Chemnitz
<b>UC</b>	SysML-based Use Case
<b>UML</b>	Unified Modelling Language
<b>VC</b>	Version Control
<b>VHDL</b>	Very (high speed integrated circuit) Hardware Description Language
<b>VP</b>	Virtual Prototype
<b>XML</b>	eXtensible Markup Language

# 1 Introduction

Development of an Application Specific Integrated Circuit (ASIC), as with developing many other electronic systems, involves a complex process of gathering requirements (“Requirement Elicitation”) for the system under consideration. The Deutsches Institut für Normung (DIN)) defines “Customer Requirements” (CR) and “Functional Specification” (FS) with respect to products requirements. Specifically, CR and FS refer to “das Lastenheft” and “das Pflichtenheft”, respectively [1]. The customer provides the CR - sometimes in conjunction with the product developer. The CR describes what the system should do. The FS is provided for the product developer, by the developer, based on the CR, stating how the product is to be realised. FSs are realised after qualification of the CR, clearly outlining the state of technology - current capabilities and limitations.

Stakeholders in a developmental project have to agree to the contents of the requirements. Stakeholders are individuals or groups who can directly impact the project and its outcomes. They have direct interests in a project’s requirements and have decisions to make towards the realisation of a project. They also include people or entities who have to work with the requirements, whether to design or implement parts of the system, the whole of it. Those who fund the project development are also members of the stakeholders.

When an agreement is reached between the customer and the system developer, as to what system is to be built, the developer continues to the next step in the development cycle, depending on the process model (V-Model, Waterfall, etc.) being employed. As already stated above, this phase produces the FS.

The ASIC architect<sup>1</sup> is often tasked with the responsibility of fulfilling the CR elicitation phase of the development. Through architectural design, functional and system requirements analyses phases, the FS is produced. This is then translated into “Module Specification”, for module developers to design and build models to satisfy the given specifications.

In the course of the module development, certain new requirements are generated and communicated back to the architect. Some deviations from the original specifications might also occur, and must be documented, and communicated to the customer, for a potential review of the system specification. There is, consequently,

---

<sup>1</sup>This role is fulfilled by different persons or groups, depending on the respective organisational structure of the entities involved.

a constant back-and-forth process with the communication of requirements between these three hierarchies of system stakeholders. This is the ideal case.

In reality, though, the communication between the architect and the developer leaves a lot of gaps unfilled. Some contexts of the underlying reasons for a design specification are not available. The automotive industry is multi-tiered. The Original Equipment Manufacturer (OEM) might pass a set of requirements down to the first level supplier (Tier-1), who extracts some specifications and contracts their fulfilment out to the second level auto-domain player (Tier-2), etc. These different players are often protective and cautious of their intellectual properties. They give only as much information as they believe is sufficient for the other party to deliver the contracted products. The contexts of the requirements are, therefore, lost in transition and translation.

The result of this communication gap is inconsistencies of requirements. The contexts missing or not provided; those supplied by the module developer, but were not communicated back. In the case of different organisations: the caution applied to requirements communication, and the resultant mischaracterisation and misrepresentation of the requirements. These are all the challenges faced at different organisations and tiers. When these inconsistencies have passed through the system development process for long and after several reviews, without fully pointing them out and addressing them, they are found in certain areas of the documentation, and are not traceable to their origins or reasons.

The voids filled by the developer, Tier 1 or 2 is usually informed by the need to continue with the project development. This often creates the inconsistencies mentioned above. Oftentimes, due to the missing contexts or miscommunications, the architect and the module developer have different understanding of the system. This misunderstanding are often not known, until much later in the development. When such misunderstandings exist, and the contexts and gaps were filled by the developer, delays may occur. These delays are often due to requirements added later in the development that were not captured earlier on. This could lead to delays in the time-to-market (TTM).

### **1.1 Motivation**

Due to the high demand for safety-critical, highly intelligent systems, more so in emergent applications like automated driving and adaptable systems, ASICs play an even bigger role in the world today. Consumers demand more autonomous, efficient and intelligent functionalities, which in turn means OEMs require more from their parts suppliers to meet and satisfy the demands placed on the market.

The architect and the developers working on the ASIC modules are often in constant communication. This helps to ensure that contexts that are necessary for

development of high quality ASIC products are guaranteed. However, meeting the demands of the changing landscape vis-à-vis interconnectivity of devices, especially in the automotive domain, requires a different approach for this communication.

In all systems development domains, the communication approach has to be more effective and efficient, in order to improve the development process. This can be in many different formats. For example, better mode of communication between the architect and the developer can help reduce latency in communication. The mode of communication might be changed from direct meetings to documents or message transfer between them, for example. The need for robust and adaptive ways of handling the complexities of the requirements and specifications, and adapting them to the ASIC development processes and workflow is, therefore, an interesting topic of research across multiple domains. Furthermore, providing the system developers with the right processes and tools can help to improve the development and manufacturing outcomes. It can also contribute towards maximising the other resources, least of all, the effort put in to realise the products.

Mich et al [2] observes that, the volume of requirements written in Natural Languages (NLs) is usually 79%. An on-going research that aims at an early formalisation of ASIC specification is at an advanced stage. This thesis is a part of that research, and its contribution will be towards the handling of the requirements, in order to engender consistency of the requirements throughout the lifecycle of the ASIC system. The specification to be formalised is currently NL-based description of the composite hardware-software (Hw-Sw) ASIC device. The requirements for a single ASIC are in the range of a thousand or more (1000+), and always vary from ASIC to ASIC. Some ASICs have similar requirements to others, with minor but significant differences. It is therefore, more time efficient and productive to write the requirements once and reuse or inherit them for subsequent ASIC development where applicable.

The challenges of handling requirements is not peculiar to ASIC development. Indeed, it is common to many other products development. Especially, with increase in requirements, comes increasing complexity of the handling and processing the requirements. Aberdeen [3:2] 2006 survey highlights that business pressures increased by 47%, when the CR increased by 43%. While it is good for businesses to have an increase in their customer needs, which directly means more production and more business, the rate of the increase in pressure does not necessarily equal in measure of productivity. This could force businesses out, when competitors take over their customers. In that survey, developers responding by a 42% increase in development activities and a 49% increase in the quality and/or performance of their products shows they responded well to the pressure [3:2]. It follows, therefore, that to remain competitive, product manufacturers have to adapt their methods and processes to a more evolved measure of staying ahead of the increasing customer demands by doing better and maximising inputs.



Considering the challenges mentioned above, better development processes that can handle large-scale production with ever-increasing complexities are always sought-after. This thesis seeks to provide a set of principles that could be implemented to reduce and eliminate the inconsistencies in requirements caused by miscommunication between the stakeholders. Of particular interest in this work is the communication gaps and miscommunication between the architect and the module developer, and the resulting inconsistencies in the requirements. The result of this thesis should be procedures that can be integrated into ASIC development process. These procedures should be reducible into steps of achievable targets that can be implemented as methods.

While the reduction and elimination of inconsistencies in requirements is the specific focus of this thesis, the work was sectioned into research areas. Research questions were developed from these areas to aid a comprehensive overview of the research. The discoveries observed in the course of answering these questions provide solutions to the research questions. These solutions are characterised by the specific processes already in place<sup>2</sup>. Therefore, they serve as a series of methods that will be practised, to improve on the development process already being practised. The research questions are described below as goals and objectives.

### 1.1.1 Aim and scope

This thesis considers the requirements that are gathered as part of the development process. The processes involved in handling the requirements are evaluated from a module and communication point of view. Communication in this context is, particularly, between the architect and the module developer, as mentioned above. An already existing ASIC development workflow is also considered.

The methods will be related to the implementation, monitoring, appraisal, documentation, cataloguing and archiving of requirements at a named environment with certain norms and guiding principles of development and production. The organisations providing this environment is made up of different business areas and engineering units, as well as product teams, that are integrated in the development and management of ASIC production. One ASIC unit would normally involve personnel of different departments and units at various levels. This means that, they all get involved with the interpretation and management of the requirements for the respective ASIC. Requirement traceability to assist these teams is expected at the end of this thesis. The solution also helps in the workflow process that will realise a promotion of early standardisation for formally verifiable models.

The goal of this thesis is to answer the following questions:

- Is requirements traceability in ASIC module development possible?

---

<sup>2</sup>The aim is not to completely overhaul the development process in place. Rather, to contribute to an emerging process, within the limits of availability.

- Does it improve communication among the stakeholders?
- Can it help provide the missing contexts of requirements?
- Does it improve efficiency?
- Will it spawn a more pragmatic development process?
- Will the process be better managed?

### 1.1.2 Research objectives

ASICs are complex hardware and software devices, which are increasingly relying on software for their functionality [4:323] [5:166] [6] [7] [8]. More software is being deployed to embedded systems, especially a network of distributed embedded systems. In the automotive domain, Electronic Control Units (ECUs) are increasingly replacing the mechanical functions of car parts. ASICs are integral parts of ECUs, which is at the heart of the innovation and progress made in the automotive industry today. The continued progress of ASIC development has to match, or even, surpass the rate at which they are deployed and implemented in cars. Wastage in system resources have to be reduced; manual processes have to be as reduced as possible; automation employed where feasible; and repetition must be eliminated. These are changes necessary for growth of ASIC development. Consequently, methods have to be adapted to ensure these changes. Realistic targets have to be set and measures to meet required goals and targets defined. Tools have to be upgraded and adapted to support the required and desired changes and upgrades to development workflows.

While the goal of this thesis creates a couple of questions to be answered, the objective creates some of its own to help answer those coming from the goal. These questions can be summarised as follows:

- What are requirements?
- What is requirements traceability?
- Is traceability specific to a particular systems' domain?
- Are there standard practices?
- If yes, how can other solutions be improved or adapted?
- How can it be used in ASIC development?

The focal point of this research work is on the development of actionable steps to observe in order to promote MBSE in ASIC development. At the end of this thesis, the answers to these objective questions would have been used to provide relevant answers to the more subjective ones posed by the goals. The theories of requirements, and how they are fulfilled will be reviewed, and the answers will be logically structured and comprehensible.

## 1.2 Research approach

To ensure a holistic approach to the work, which implies understanding the requirement gathering and processing activities, as well as documentation, cataloging and actually working with the requirements to realise an ASIC product, certain methodologies were employed. A couple of academic methodologies that serve as a framework for the systematic development of theories and applied knowledge quickly come to mind. After review of some of these methodologies, two of them - Action Design Research (ADR) and Questionnaire-Based Gap Analysis (QBGA) - were applied, and are briefly described here.

### 1.2.1 Action design research

According to Wieringa & Morah [9:220], action research stems from social sciences, whereby intervention is sought in order to “experimentally” improve on social conditions, while also learning from the existing condition. It involves scientists analysing and “diagnosing” the outcome or state of a social event, occurrence, or situation, and proffering alternatives or solutions, while learning from the (existing) occurrence. Action research regards the theory of research as “based on tentative ideas”, which are then improved upon after cycles of application of arising new ideas [10:54] [11:40].

This approach to evaluation of events and occurrences, their theories of evolution and applied solutions to problems has been evaluated in the area of systems and software design analysis and development [12].

ADR is an extension of action research. It is aimed at prescribing a constant evolution of “design knowledge” by acquiring and evaluating the knowledge inherent in a system of “assemblage of information technology (IT) artifacts” in the course of using a particular design knowledge base and approach for systems development [10:54]. This is the case where there are already existing theories and technologies to build upon, as is the case in the project area this thesis is based. In essence, it is not typically iterative in a cycle, but over time, the same systems development process is improved, refined and reused, and the knowledge gained from the preceding installments of the same development processes are reinvented and applied to succeeding developmental process.

### 1.2.2 Questionnaire-based gap analysis

On its part, QBGA requires the use of questionnaires to collate data and information in order to evaluate the state of the system, as perceived by those who are currently working with it [13]. Questionnaires are very valuable in obtaining information from those who have first-hand knowledge of the system. Closely related to questionnaires are interviews. They differ in their inherent nature: interviews are more interactive;

they require directly asking the questions, and explaining the question where necessary, whereas questionnaires require submitting the written questions. Due to the organisational setting and the proximity and accessibility to the just about enough of the necessary resource persons to engage using this method, the gap analysis method used each time it was necessary was interview-based.

Interviews were conducted during the course of this work, rather than questionnaires. This also helped in contextualising, not only the questions that were posed to, but also the responses that were obtained from the persons involved. Figure 1.1 is a graphical representation of the research methodology, and is a nesting of both methodologies reviewed above. It can be observed from the figure that, some of the steps are iterative, while maintaining structure and focus on the target end.

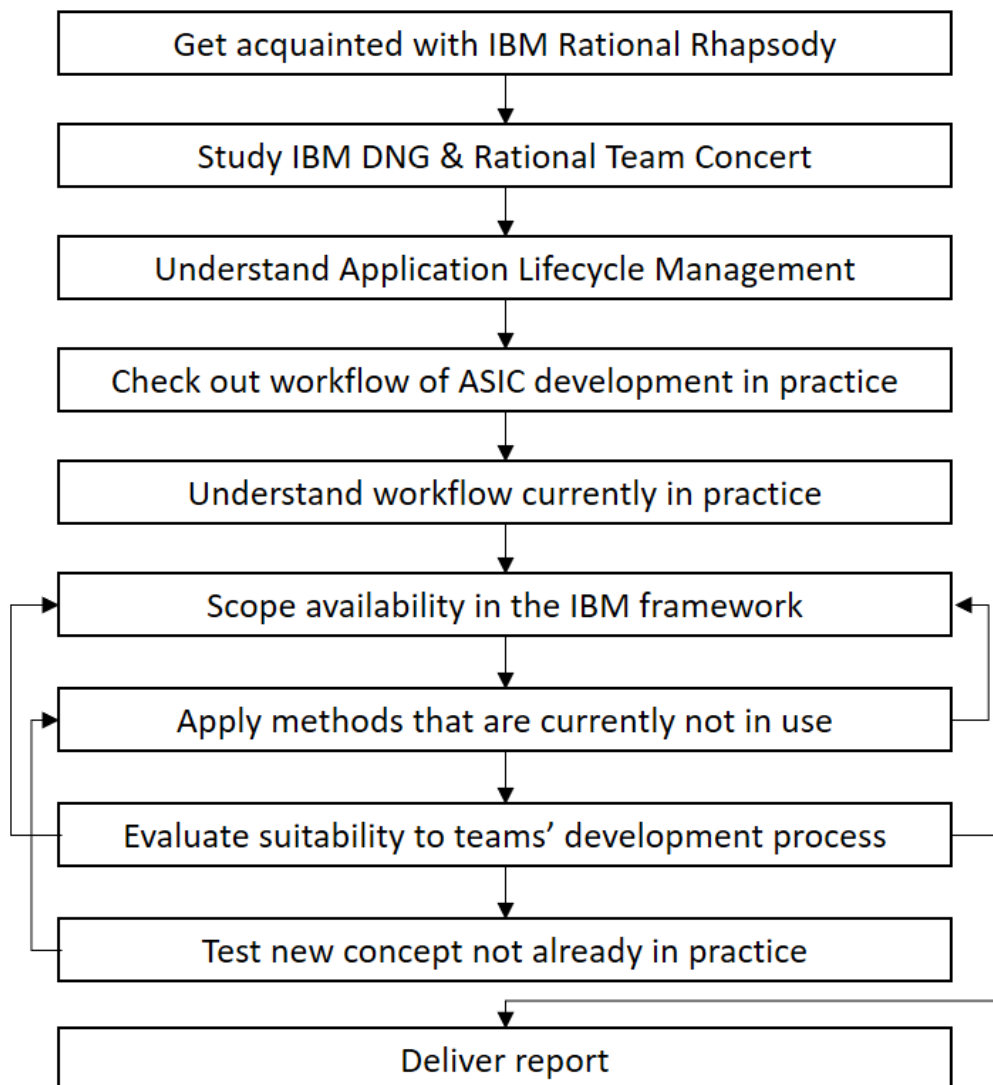


Figure 1.1: Research methodology

## 1.3 Professorship of Computer Engineering at the Technische Universität Chemnitz

This work is registered and supervised internally, at the Professorship of Computer Engineering within the Faculty of Computer Science of the Technische Universität Chemnitz (TUC), chaired by the Prof. Dr. Dr. h. c. Wolfram Hardt.

The professorship is renowned for the on-going and continuing research activities in the area of embedded systems [14] [15]. The research areas include the integration and applications of hardware and software in complex use cases, some of which include: the role of embedded systems in car-to-car (Car2Car) and car-to-x (Car2X) communication, reconfigurable systems in/and self-organised network systems [16]. Further research areas include adaptive flight control algorithms and image processing in such specific areas as their “Adaptive Multicopter Mission” for use in special missions, like its application to the monitoring of power lines and buildings [17].

One of the most visible sights in the Automotive Software Engineering (ASE) laboratory at the TUC is the “YellowCar” project developed by the professorship [18] [19]. This project uses multiple ECUs connected via controller area network (CAN)<sup>3</sup> as a platform to perform functional tests of functional automotive application units. It also uses multiple sensors for these units. Such This can be applied for advanced driving assistance systems (ADAS) functions, for example.

The professorship has been teaching and impacting knowledge in these specific areas through courseworks like Hardware/Software Codesign 1 & 2 and Hardware Development with VHDL<sup>4</sup>. Furthermore, with its expertise in automotive-specific domain topics like AUTOSAR<sup>5</sup>, the professorship also offers students the opportunity to learn theories, and gain hands-on practical knowledge and experience, of AUTOSAR-related topics, such as laboratory work with ECUs. This is done through courses like ASE<sup>6</sup>, Seminar Topics on AUTOSAR Based Software Design, etc [20].

Formal Specification and Verification (FSV), one of the Winter Semester courses available at the professorship, is of particular import. It focuses on the formal specification, validation and verification of (distributed) embedded systems using methods like Temporal Logic (TL), including model-checking with such tools as Simple Promela INterpreter (SPIN), a PROcess MEta LAnguage (Promela) based tool. FSV coursework also provides opportunities for hands-on knowledge development, with examples of formal verification methods from aerospace and automotive industries.

---

<sup>3</sup>CAN is one of the prominent network communication protocols used in automotive.

<sup>4</sup>VHDL is an acronym for Very (high speed integrated circuit) Hardware Description Language

<sup>5</sup>AUTOSAR means AUTomotive Open System ARchitecture

<sup>6</sup>In the ASE Study Programme of the TUC, a particular laboratory-based coursework is also titled ASE

However, the choice of the professorship is informed by its vast experience and expertise in the area of hardware software codependent systems, particularly with Field-Programmable Gate Arrays (FPGAs) and ASICs [21].

### 1.4 Partner organisations

Automotive Electronics (AE), a division of the Robert Bosch GmbH (Bosch), is in the business of semiconductors, which includes the responsibility of designing, developing and manufacturing ASICs for both internal (Bosch) and external customers [22]. The business areas covered by Bosch are categorised into four sectors: 1) Mobility Solutions, 2) Industrial Technology, 3) Consumer Goods, and 4) Energy and Building Technology [23]. These include areas like smart and connected homes, electronic-bicycles (E-Bike), artificial intelligence (AI), etc.

As they write on some of their internal posters, “We are not always visible, but we are always there”. This is very informative, because very few people outside of the automotive sector know that Bosch is one of the foremost developers and suppliers of products used in automotive systems. Many people associate Bosch with the products that are always visible: washing machines, refrigerators, microwave ovens, electric irons, etc. These are household consumer products.

AE is a part of the “Bosch Mobility Solutions” sector. Indeed, as of the year 2019, Bosch is the world’s leading supplier of general automotive products [24]. It is also ranked as the world’s sixth (6th) largest manufacturer of automotive semiconductor for the same year [25]. Worldwide demand for semiconductors as of the year 2018 was 11.5% [26]. This means that Bosch, and by association, the AE, plays a vital role in the value-chain of global semiconductor supply. As already mentioned in Section 1.1.2, ECUs are dependent on ASICs and ASICs are dependent on semiconductors. This research is in line with the vision and action that has driven the development of ASIC at Bosch.

This work has been developed in the ITEA3 project COMPACT (reference number: 16018). COMPACT is funded by the Austrian Research Promotion Agency FFG (project number 863828 and 864769), the Finnish funding agency for innovation Tekes (Diary Number 3098/31/1717) and the German ministry of education and research (BMBF) (reference number: 01IS17028). The authors are responsible for the content of this publication.

### 1.5 Structure of this thesis

This thesis is divided into seven chapters. Chapter 1 discusses the challenges involved in the communication and consistency of requirements. It also discusses the aims and objectives of this research.

Chapter 2 takes a brief look at some background technical concepts of ASIC development as a function of requirements. It also details some standardised systems used in this work.

As this thesis is specific in its subject matter, Chapter 3 will look at the state of the art from the view point of standards that deals with this subject, in terms of modelling and their transfer. The chapters concludes with a general overview of some of the challenges and solutions in practice.

Chapter 4 introduces conceptualised solutions as ideas for providing solutions to the research questions. It is more like a brain-storming session of one that narrows the focus for the rest of the work down to the specific research objectives.

In Chapter 5, the specific capabilities and properties of the systems and tools that are building blocks to the solution are introduced and used. The methodologies that are expected .

Chapter 6 applies these capabilities and properties to a real-life development process in the form a case study. The real-life ASIC project is used to evaluate the case study as a proposal of methodologies.

Chapter 7 chronicles this work as summary of a set of questions and matching answers with respect to the emerged methodologies. It also makes recommendations for future related work, in furtherance of the new methodologies introduced.

### **1.6 Summary**

In this chapter, the automotive sector was introduced as a multi-tiered industry. The complexity of requirements as an inherent challenge due to this multi-tiered nature of the sector was also introduced. This chapter also introduced the requirement inconsistencies that occur as a natural consequence of the complexity of requirements and of the nature of multi-tiered industries. The organisations with specific interests in this work were also introduced. The chapter ends with a brief description of the structure of this work.

## 2 Technical background

This chapter briefly discusses ASIC and some of the technical concepts that characterise the complexities of its development and production. Requirements and systems engineering are also discussed. The chapter ends with the introduction of languages as a tool of requirements and systems engineering.

---

Semiconductor technologies have changed the world of computer science in particular, and the world at large in such a rapid fashion [27]. Indeed, it is still changing and shaping the technological landscape. By the end of 2018, the automotive sector accounted for 11.5% of the demand for semiconductors worldwide [26]. These semiconductors are mostly used in the manufacture of ASIC. And ASIC is used for the production of ECUs. For example, the use of smart-sensor ASICs as signal processors and for the preprocessing of sensor signals, as well as the provisioning of data for ECU microcontrollers.

### 2.1 Application-specific integrated circuit

Application-Specific Integrated Circuits (ASICs) are composite Hw and Sw embedded system components. They have “signal processing capacities” due to the presence of “micro-processors/-controller cores and memories”, as well as “analog interface and pre- and post-processing circuits” [4:321]. They (can) also contain wireless interfaces, the number of which keeps increasing [4:321]. The computing functions of ASICs are implemented using their signal processing abilities. The micro-processor/-controller in the ASIC is mainly the control unit, while the program code and data samples are contained in the memory [4:321] [28:283]. They are designed for customer-specific purposes, e.g. processing of sensor-received signals of airbags in a vehicle, triggering the deployment of the airbags [29] [30] [31].

As the use of artificial intelligence and advanced networking in cars increasingly drives the growth of the automotive domain, more software is being deployed in the design and implementations of the functions performed by the ECUs, as mentioned in Section 1.1. For example, ADAS depends on advanced functions implemented on ECUs. So does Internet of Things (IoT) implementations, like in parking and infotainment systems<sup>7</sup>, etc. Functions like these are some of the driving forces behind the growth of ASIC development in the automotive domain. Furthermore,

---

<sup>7</sup>“Infotainment” system is coinage to represent informational and entertainment system, like audio-visual and navigational systems.



the complexity of ASICs keeps increasing with the continuous increase in the size and volume of software that is used to drive ECU functions [4:323] [32:2] [33:1]. And, as transistors continue to shrink, and processors get even more complex according to Moore's Law, ASICs have evolved into Systems-on-a-Chip<sup>8</sup> (SoC) and, more recently, Systems-in-Package (SiP) [4:104, 519] [34:1] [35:30] [27].

SoCs are single chips, onto which multiple functionalities needed for a system are loaded. They contain one or more: micro-processor/controller cores, memory systems, input and output (I/O) systems, analog-to-digital and digital-to-analog converters (ADC & DAC) on the single microchip.

SiPs, on the other hand, contain multiple microchips stacked atop one another, thereby avoiding the use of multiple spaces on a printed circuit board (PCB), as in the case of a SoC. SiPs is a packaging technology used in semiconductor world to circumvent Moore's Law.

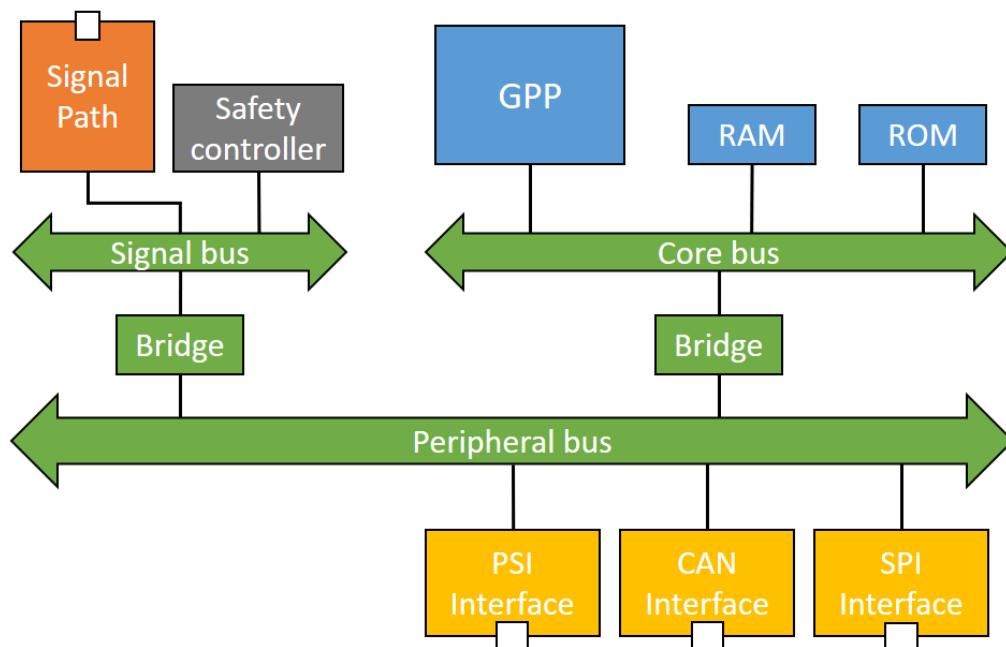


Figure 2.1: Example of on-chip Architecture of smart-sensor ASIC [36]

Figure 2.1 shows an example simple architecture of a signal processing ASIC. It consists of a general purpose processor (GPP), a read-only memory (ROM), random-access memory (RAM) for tasks like boot loading; a safety controller and a signal path component for signal processing related tasks; and connections to external components through the serial peripheral interface (SPI), CAN interface and program structure interface (PSI). These different components are connected with a system of bridges and busses.

<sup>8</sup>Some sources also refer to it as Systems-on-Chip

The relationship between the phases of hardware and software components in ASIC development can be visualised in Figure 2.2. It can be seen in Figure 2.2(a) that, the traditional workflow of the development process involves the availability of the ASIC device, before the commencement of the development of the software to be implemented on the ASIC. This Hw can be developed as a virtual prototype (VP) of the finished ASIC Hw device using FPGAs [36]. Figure 2.2(b) shows the workflow, where the use of a VP ensures the development of the Sw components for the finished ASIC device at a much earlier time, as compared to the traditional workflow [36:4] [37]. The serial development process in Figure 2.2(a) is bettered by the pseudo-parallel process in Figure 2.2(b). Consequently, the TTM of the particular ASIC is shortened as in Figure 2.2(b), which is the current workflow.

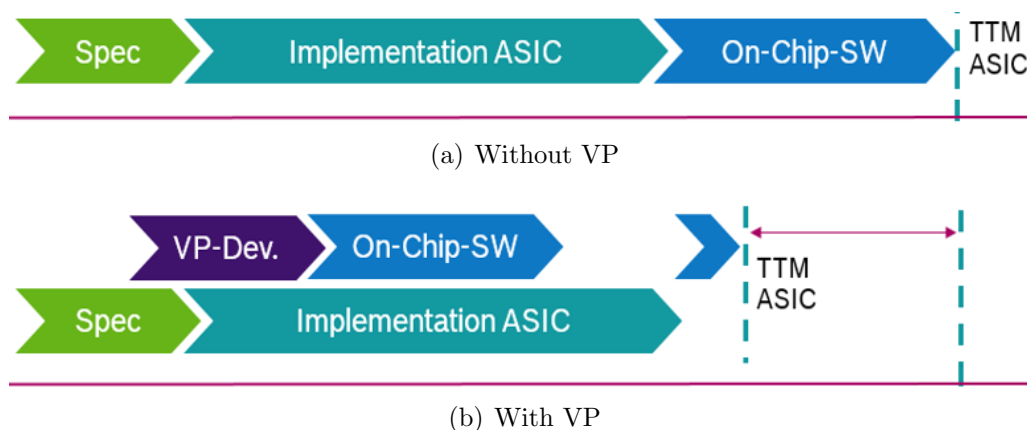


Figure 2.2: ASIC development workflow [36]

The ASIC components described above are, oftentimes, built by different organisations and assembled as a unit by some other. They are often built from already available components, too. The workflow described might also involve different organisations, due to the complexity of the supply-chain in the automotive sector, as described in Chapter 1. There is therefore a need to reuse already designed module components as much as possible. Reusing modules means that, a module used for a specific ASIC are re-adapted for some other similar ASIC, or for similar specific functions in a different ASIC. This reusability is often in the form of intellectual properties.

## 2.2 Requirements engineering process and management

There are many definitions for Requirements Engineering (RE), but one that frames it in alignment with the characterisation of this work is:

*“RE aims to discover the purpose behind the system to be built, by identifying stakeholders and their needs, and their documentation”* [38:6].

It names RE as being a process. A process, as is related to here, is an ordered set of actions performed systematically to achieve a named outcome. It is a coordinated set of ordered and systematic activities that leads to definite result. So, with the process nature in focus, there are inputs into the process. These inputs include the use cases, contexts, expectations of the customers, and the decisions that are made to achieve the set targets.

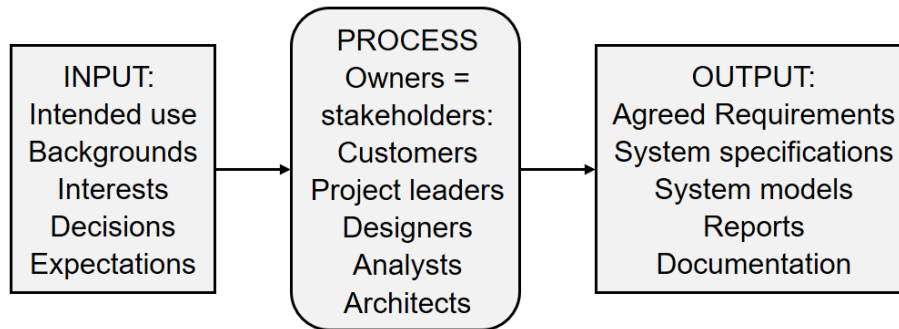


Figure 2.3: RE as a process

The outputs of the process are the results produced from the activities that are carried out in the process. These outputs include the requirements that are agreed upon from the gathered use cases. They also include the system specification, the models that fulfil the agreed requirements, the reports, and the documentations that are produced as part of the RE process. Figure 2.3 shows the process nature of RE, which also depicts examples of its inputs and outputs. The process is also shown to be owned by stakeholders (Figure 2.3). Stakeholders have already been described in Chapter 1, as those entities who have specific interests in the project.

On its part, the set of ordered activities put together to form a RE process is depicted in Figure 2.4. The activities start with stakeholder identification, where the requirements engineer identifies the entities, especially from the customer-organisation, who have specific interests in the ASIC to be developed. The requirements engineer here refers to the ASIC architect. The needs of the customer, i.e. the CR, are captured as use cases. After identifying the use cases, the requirements are captured, then analysed. This analysis of the requirements produces the agreed requirements. But, as can be seen from the illustrations of Figure 2.4, the clarification and restatement of the requirements follows the requirement identification phase, when there is a need to re-work the requirements. This could be in a situation where there is a conflict in the requirements. Analysis of the requirements also leads to classification and definition of terms for handling the requirements. Classifying and defining different terms associated with the requirements ensures each word applied to a requirement conveys the same meaning to each stakeholder. This classification and definition of terms could be interpreted as a naming convention.

Requirements are further prioritised, derived, partitioned and assigned to the developers for the development of the models. These steps come after the specification of

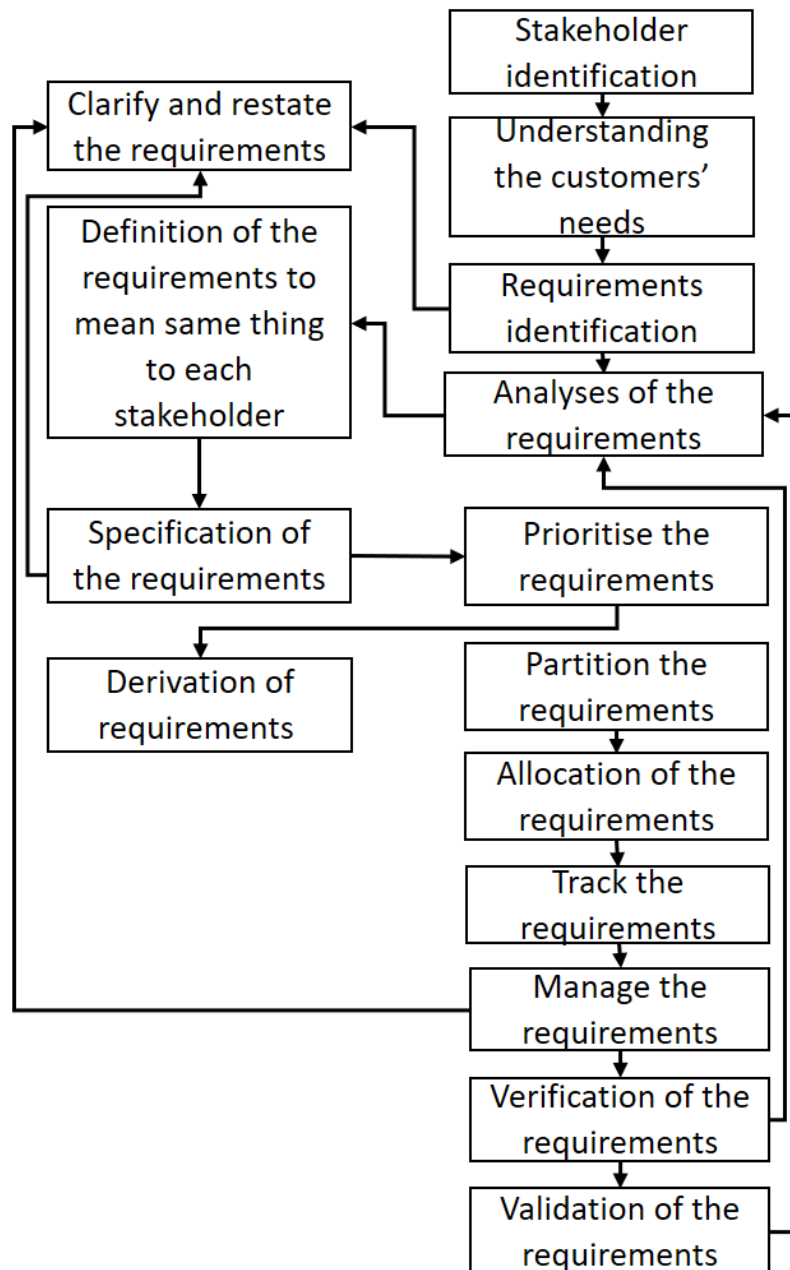


Figure 2.4: Workflow in the RE process

the requirements. Figure 2.4 also depicts the tracking of requirements. Requirement tracking is supposed to be started immediately after it has been captured to ensure its consistency across a project lifetime<sup>9</sup>. Tracking is also part of the process during the development of models that fulfil the requirements.

The requirements that are being put into effect in a product development also has to

<sup>9</sup>This thesis is based on the tracking and traceability of models to requirements

be managed. This is captured in Figure 2.4 as “Manage the requirements”. Requirements management (RM) is a subset, but also, an integral part of the RE process. The requirements are managed according to prescribed norms and standards that must be observed in a project development. To guarantee the development is based on requirements that meet the prescribed norms and standard, the requirements are verified and validated against those norms and standards. This step of the RE process is also captured in Figure 2.4. Figure 2.4 is an adaptation of “Requirements Activities in the System Life Cycle” section of Young [39:4-5].

To design and develop real-time embedded systems, a process has to be engineered that enables the stakeholders to manage and trace the requirements precisely [8:5]. There are standards and norms that must be adhered to during these development processes. Some of such standards that inform processes the must be critically adherent to them are the IEC<sup>10</sup> 880 in nuclear systems development, CENELEC<sup>11</sup> EN<sup>12</sup> 50126, EN 50128 and EN 50129 in railway systems in Europe, and DO-178<sup>13</sup> [8:6].

The automotive sector has introduced a standard for safety of road vehicles known as the International Organization for Standardization 26262 (ISO 26262) [42] [43]. ISO 26262 is a norm which requires a safety approach to development of electrical and electronic systems intended for use in road vehicles. This approach stipulates that the development of such electrical and electronic systems be consistent and concerned with the safety of use of the parts in all levels of development [36:6]. This norm ensures that, at every stage of the development of vehicle components, especially the safety features of a vehicle, the process in place implements safety of these parts during the development. In essence, the requirements being collated must be informed by safety concerns.

The question of what requirements are have been grappled with by scholars. Definitions keeps evolving, but considering the definition by Young in [39:1], a requirement is an attribute required of a system - a feature that the system should possess after it is produced and delivered - written as a statement that identifies what capabilities the system must possess, be it a quality and/or capacity the system must have in order to add the intended value that is conceived of the system before production. Albinet et al [8:5] considers requirements as having “a structure with several attributes” and “characterised by an identification, a textual description”, but also could be of either a functional or non-functional type. While the functional requirements of a system might be more interesting to developers, non-functional requirements (e.g. cost, safety, performance) must be captured and documented, because they are to be managed and maintained properly, in order to realise a project. The overall

---

<sup>10</sup>IEC is an acronym for International Electrotechnical Commission

<sup>11</sup>CENELEC stands for European Committee for Electrotechnical Standardization

<sup>12</sup>EN stands for European Standards

<sup>13</sup>DO- refers to Document [40][41] and DO-254 in aeronautic system”

well-being of a project is determined by the concord between the functional and non-functional requirements, as it is between both the developer and other stakeholders (customer, project manager, financier, etc.). Attributes like performance can also directly be related to, and linked from functional requirements like voltage (voltage is a functional requirement, and can be modelled).

It has been said quite often, in one way or another, that requirements are volatile [38:3] [44:64] [5:151] [45:594]. RE is a practice that deals with the elicitation and analysis of users' requirements, then specifying and verifying them, before transmitting them into products requirements in a repeatably systematic way, to ensure they are complete and consistent with the requirements that were elicited at the start as well as the specification that was born out of them [46:161]. RE helps developers, engineers and architects of systems to cope with the volatility of systems design, by providing functional and non-functional behaviours a system should exhibit, to ensure validity and consistencies with standards and specifications [47:271].

Words have different connotative and denotative associations. They are liable to changes and differences in their contexts and meanings. These differences also affect how they are understood. It is within the realm of RE to capture those changes and differences. The connotative and denotative associations of words used in RE present the "context" of the requirements. Many procedures have been reviewed by scholars over the years. The approach by Prakash and Prakash [38:7-8] best describes the workflow within the scope of this work, and is described below.

### 2.2.1 Requirements elicitation

The requirements elicitation phase of systems development involves the entire process of gathering the information necessary to conceptualise the system. It ranges from what the users of the proposed system want to have in the system, to what the management wants or do not want as part of the system's capabilities. The requirements engineer, at this point, wants to understand the vision the stakeholders have for the system. Outlining what the likely constraints of the system are, depending on the varying perspectives of the stakeholders is included in this phase of the RE process. The contexts the conceived system will be deployed into is an important part of this phase. This is, in part, to determine whose opinion of the built system is required in order for the system to be accepted [48:27]. This step includes, identifying what the system should be capable of doing from the customers, working together with those who have direct knowledge of what the system should do.

The requirements are prioritised, and the technical know-how of systems' module developers is used to prepare a draft of the system architecture. It is also necessary to identify the domain experts with respect to the context of the system, as well as related and useful literature, and any existing or required software or system related to the conceived one [48:27] [39:17]. The result of this step is a realisation of an overview of the system from the user's perspective, as well as potential

external influences on the intended system, and a picture of “*stakeholders’ respective backgrounds, interests, and expectations*” [48:27], in what is described as the “stated requirements” by Young [39:1, 50, 62]. This step is also, partly iterative, because requirements keep being modified, and the back-and-forth already described in Section 2.2 above comes into play, hence the tag, “Volatile”.

### 2.2.2 Requirements analysis

After gathering the requirements, which falls into the category characterised as “knowledge problems” [49], the “real requirements” is to be deduced [39:50]. The classification as real requirements represents the realistic portions of the use cases captured from the CR. An impression is made, about what the system means to the different stakeholders. This impression is to put the system context in the stakeholders’ perspective, and to know what they want the system to do. This also factors in all possible known effects of the system operation to its environment. The real requirements is the result of analysing the stated requirements that has been gathered during the elicitation phase [39:1, 50, 62]. The architects (in this case) communicates and collaborates with all the other stakeholders - the customers, designers, management, etc. - to present the outlook of the proposed system as it fits into the conception of each stakeholder. This leads to even more understanding of the intended system and what the system would be able to do, or not do, when it is completed. Here, the state of technology is reviewed by the designers and developers, and communicated to the customers.

### 2.2.3 Requirements specification and documentation

After collecting the “real requirements”, as a collaborative effort between the system designers and customers, these requirements are formally specified, resulting in the FS of the system. This phase of the process involves application of terms defined during the first two phases. It also includes resolving conflicts of requirements where they occur, and often leads to the discovery of more requirements. Depending on the discovered requirements, the analysis or the negotiation phase may have to be revisited. If any changes are to be made, the stakeholder, whose view of the project is affected by the change has to be informed. The project continues after this change is noted and the awareness about the change communicated to the require stakeholder. However, not all changes are communicated to the customers. Some communication are between the architect and the developers who derive new requirements from the already available technical details.

Concurrent to the specification activity, is a very vital step: documentation. It is important to have written every detail in the previous two phases down. The critical nature of the FS documentation is such that, the specification is standardised, and is to be referred to during the development and maintenance of the system. The FS is the precursor to the system design phase, therefore its documentation is critical.

Subsequently, consultation of the CR is only to inform the contexts of requirements during implementation. The documentation contains the technical details of the system to be built. The technical details are archived at the end of the development of the system.

### 2.2.4 Requirements verification and validation

As already mentioned in Section 2.2, requirements are verified and validated against norms and standards. This is a verification and validation of the requirements (not of the system). The documented requirements have to be checked against standards and norms observed by the guidelines that govern such systems. They also have to fulfil the customers needs. The specification is verified and validated for “consistency and completeness” [38:8]. This step involves the project managers and the architect. The requirements engineer - in this case, the architect - and the requirements managers discuss the requirements against the governing norms and standards. The architect has to justify the reason for the requirement as specified, and the manager has to decide whether it meets the norms and standards. They also have to ensure the customer’s needs are met by the specification. This process is implemented through the following activities described by Young [39:4-5], as described in Section 2.2 (see Figure 2.4). The activities are implemented by the use of methods, systems and tools. There are many tools and methods that are deployed in the RE process. The aim of RE is to evolve a method of using the systems and tools available to the best of their capabilities, according to norms and standards that govern the processes.

## 2.3 Systems engineering

The previous discussions in the sections above are building blocks to the discussions that follow. A pertinent question to ask is, “what is systems engineering (SE)?”. But firstly, when one is referred to as a “systems engineer”, what does it mean? Some of the concepts presented by the International Council on Systems Engineering (INCOSE)<sup>14</sup> help answer these questions. According to the INCOSE, systems engineers and their activities are at the fore-front of the creation of “successful new systems”. Whenever the responsibility of the team members in a systems development project involve “systems concepts, architectures, and design”, the actors are referred to as systems engineers and the team is a SE team [50]. The concept of a systems engineer further encompasses those who have to analyse the system being built before, during, and after the system has been built. Additionally, the SE domain also verify the system’s satisfaction of the requirements. Systems are typically complex, as they are made up of other systems. Their design and implementation

---

<sup>14</sup>“The INCOSE is a not-for-profit membership organization founded to develop and disseminate the interdisciplinary principles and practices that enable the realization of successful systems” [50]



do follow a RE process. Systems also depend on other systems that are not part of their design. For example, ASIC development depends on the availability of semiconductors, which are processed and supplied by another set of RE process. Systems engineers are also involved in managing this complexity. They manage the risks of modules and systems design and implementation [50].

To proceed, the question: “what is SE?” has to be addressed. It is defined as an engineering approach that fulfils the “*realisation, use and retirement of engineered systems*” and system of systems which cuts across all disciplines, using management concepts, principles and processes that are integrative, methodical, scientific and technological [50]. The terms “engineering” and “engineered” are used here to show that the process and procedures are “artful” and comprising “*of any or all of people, products, services, information, processes, and natural elements*” [50] involved in the realisation of a system over the entire product’s lifecycle respectively. This follows that, all the steps taken towards the development of an ASIC are within the discipline of SE.

### 2.3.1 Natural languages in systems engineering

As mentioned in Section 1.1, requirements are mostly written in NLs (79%). These are very often open to interpretation and are often ambiguous [51:1]. This is inherent in the nature of human languages [52:198].

As at the time of this writing, engineering of systems are based on two approaches that depend heavily on spoken and written words as the means of communication. These words are in form of NLs. NLs are those written and/or spoken languages which are known to man and have evolved over the years as a means of communication. Igbo<sup>15</sup>, German and English are typical examples of NLs. The use of NLs in the processing of requirements is essential for communicating the needs and contexts of the requirements amongst stakeholders. Requirements are typically written in NL forms. And the discussions to derive and manage the requirements are had through means of NLs. The inputs into RE as a process are mostly communicated through NLs. The outputs of the RE process are documented and reported in NLs. This includes all documents that are generated. Discussions in the RE process all take place by means of NLs. Decisions made are communicated through NLs, either verbally or in written form. The entire process discussed in Section 2.2 is achieved through NL. Essentially, NL is a tool of RE.

The two approaches of SE are: 1) documents-based systems engineering (DBSE), which also known as the traditional approach to SE, and 2) model-based systems engineering (MBSE).

---

<sup>15</sup>Igbo is the language of the Igbo ethnic group of south-eastern Nigeria. The written form of the language is latin-script

### 2.3.2 Document-based approach to systems engineering

DBSE is a system of communicating in NLS through means like papers, cardboard sheets, etc. in the development of system. The method of communication also includes signs and symbols defined as a convention, according to the steps already described in Section 2.2. This approach is referred to as the “traditional approach to SE”. The tools dependent on are simple easy-to-acquire materials that can be easily deployed. The methods can also be simple methods, like the use of “free-hand” sketches, colours and shapes. The communication of the materials and methods can also be simple. For instance, the transfer of a developed document from one person to another might require the material be taken by the sender to the receiver. They might have to define a schedule of location and time where they would meet each other for this transfer to take place. Mind maps are also very useful in the discussion and brainstorming sessions during the RE process.

The traditional approach to SE also need to generate text artifacts for use. The use of word processors to generate text-based documents is also categorised as the traditional approach to SE. The text-based documents are then transferred either physically, or virtually, using electronic means. The notion of DBSE can, therefore, be summarised as the use of text-based means and methods in the systems engineering domain. DBSE also includes development of prototypes of the system, especially, the architecture of a system as a sample model. This model is physically developed and referenced to in the course of the project. This means the use of NLS as a means of communicating the systems development methods and processes.

### 2.3.3 Model-based approach to systems engineering

MBSE, on the other hand, is a development approach where the documents, shapes and models used are entirely based on formalised systems and tools. These documents are developed, generated and managed entirely as formalised computer-based artifacts. The transfer of these artifacts are also central to the concept of MBSE. While the need to communicate development information is the same as in all forms of development, MBSE focuses on the transfer of model artifacts, rather than texts. It also prescribes the central location and management of the development model data, thereby eliminating the need for “transfer” of the information from source to destination. The systems and tools used are typically specific to the type of artifact that is being developed and managed. A purely MBSE approach uses model-based artifacts, without the need for NLS [53:4]. An ideal MBSE environment would be the use of models in all the communication conventions and mechanisms.

The use of models hinges on three things, or as Delligatti [53:4] refers to it, “three pillars”, namely: 1) a language for modelling; 2) a conceptual process method or methodology to implement the models; and 3) a befitting modelling tool for the SE domain. A modelling language for the practice of MBSE is standardised for the domain. The tools that enable systems engineers practise MBSE are available.

And, although SE concepts seek the evolution, and are evolving, into MBSE, no SE practice is completely MBSE. The most predominant form is the hybrid of DBSE and MBSE [53:4]. This is because, MBSE currently also depends on the use of NLs. The subsequent sections and chapters of this work describe the progressive evolution of MBSE, especially in relation to the development cycle of ASIC.

### 2.4 Unified Modelling Language

As stated above, a modelling language is required for a model. Diagrams are uniquely suited to describing information in a manner that systems engineers across multiple or specific domains understand in the same way. The Unified Modelling Language (UML), as a standardised language<sup>16</sup>, fits the purpose of diagrammatically representing information for software-systems engineering. UML is standardised by the “Object Management Group<sup>®</sup>” (OMG<sup>®</sup>). It is used to represent parts or whole of a system and its operating environment and surroundings. It also provides a mechanism for representing the system users. In other words, a system can be diagrammatically represented, in its surroundings using UML. However, it is noteworthy that, modelling is not to capture the diagrams. Rather, it is about the idea conveyed with the diagram [53:19]. The things that affect the system and those affected by it can also be represented as a set of UML based artifacts. This includes entities used by the system and those that use the system.

UML was created to manage the complexities and risks associated with the documentation and transfer of requirements in the development and management of software projects. At the start, developers recognised the need to define standard objects for a specific project to convey the same meaning to the stakeholders during the lifecycle of the project. Due to the diagrammatic nature of UML, it is employed in the designing of object-oriented complex systems, using diagrams to represent software systems and processes in a developmental workflow.

There are three major categories of UML model elements. These elements can be used to represent the component-parts of a system as single, and as blocks of functions of the system. They can also be used to represent the environments in which the system operates. UML names these elements as “individuals” [54:12]. The three categories described by OMG<sup>®</sup> are:

**Classifiers:** these are descriptive of sets of objects with values which are identifiable by the state of the object’s properties.

**Events:** sets of possible occurrences, whereby an occurrence is representative of anything with a consequence on the system when it happens are described as events.

---

<sup>16</sup>UML 2.5.1 (current version) is available at <https://www.omg.org/spec/UML/2.5/PDF>

**Behaviours:** sets of possible executions are described as behaviours. Executions are set of actions occurring over a period of time. Changes in the states of the objects are also captured as behaviours.

UML also defines profiles, which is one of the very important concepts of the language as a standard for SE in general. UML is typically used for software engineering systems modelling. However, the UML profile is adaptable and extensible for use in many systems development domains and platforms (e.g., embedded system). In essence, profiles in UML is used to customise UML and allow for creation of domain- or project-specific objects or symbols, as well as properties or semantics that are not part of the UML standards. One of such customisations has given rise to another language that is also standardised by the OMG<sup>®</sup>.

## 2.5 Systems Modelling Language

As described in Section 2.4, UML is generally used for modelling in software development. In contrast, Systems Modelling Language<sup>®</sup> (SysML<sup>®</sup>) was created and standardised by the OMG<sup>®</sup> for systems in general. These systems are composed of parts such as hardware, software, procedure, etc. It is a defined profile of UML, with its own extensions. It is evolved from UML 2.0 and builds on the UML 2 specifications, but with its own definitions and extensions of the UML standards. It has support for modelling of processes, behaviour, requirements, etc used to integrate a complete SE process of other engineering analysis models using eXtensible Markup Language<sup>17</sup> Metadata Interchange (XMI<sup>®</sup>). It is described as “a dialect of UML 2.0” that enhances the practice of MBSE<sup>18</sup>. Figure 2.5 shows the relationship between UML and SysML as standardised languages.

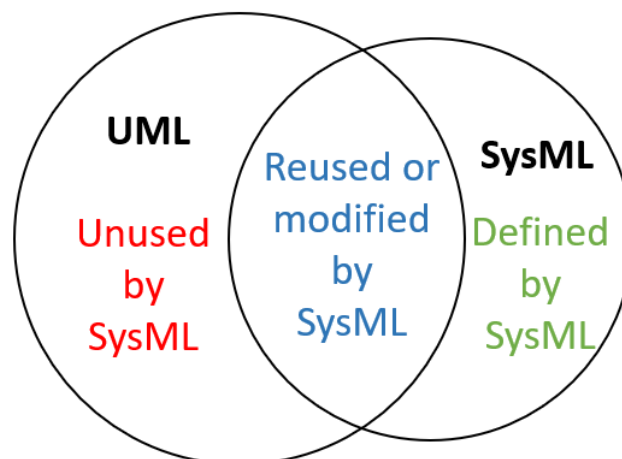


Figure 2.5: Research methodology

<sup>17</sup>eXtensible Markup Language has the acronym “XML”

<sup>18</sup>SysML<sup>®</sup> 1.6 (current version) is available at <https://www.omg.org/spec/SysML/1.6/PDF>

Most of the SysML concepts are taken from UML, and are represented as the intersected region of Figure 2.5 (marked by blue-coloured text). The figure also shows that some things are not used by SysML (identified with red text), while some are new to the standard (green). For example, while behavioural and structural diagrams are common to both standards, the concept of requirement diagram was defined by SysML. Therefore, requirement diagrams belong to the green-marked region. The activity diagram, however, is modified for SysML specification. That is, although it is available in both UML and SysML, the implementation in SysML is different from that of UML. This means activity diagrams belong to the blue-marked region.

SysML uses some of the concepts available in object-oriented programming languages in its description of items to be used for the representation in models. The concepts that are relevant in the context of this work are described below, and are illustrated using Figure 2.6.

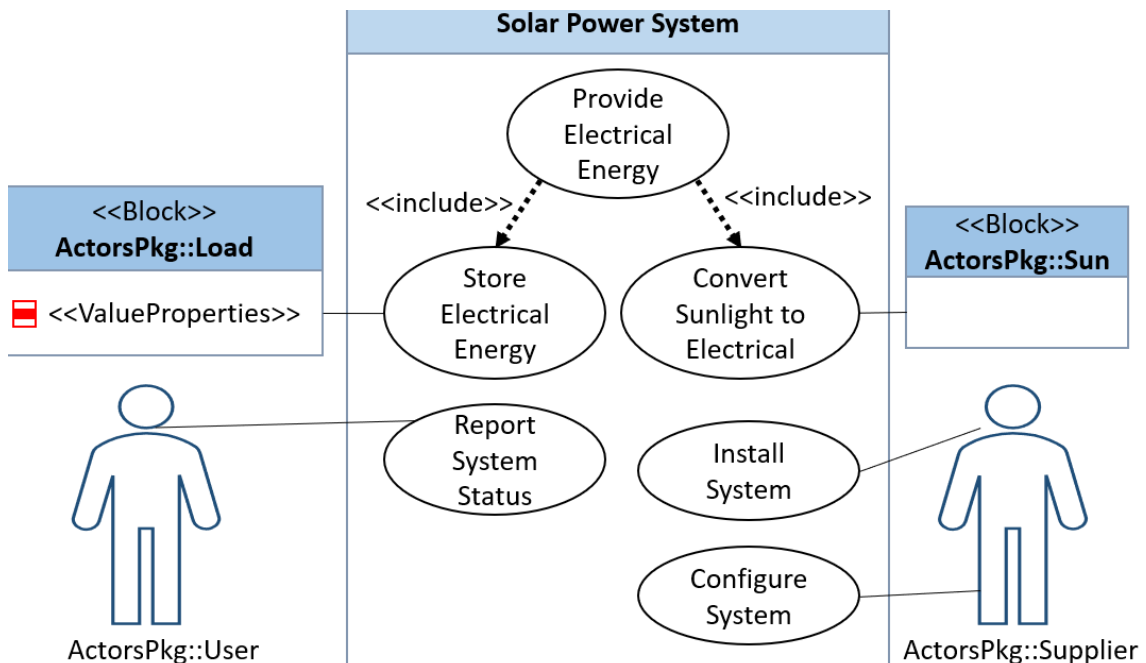


Figure 2.6: Sample of a SysML-based model

**Block:** A block describes a set of objects. It is a classifier, with the objects having the same features, constraints and semantics. The attributes can be behavioural or structural, and are associated with operations. A constraint can be a precondition or restriction imposed on an element or a set of elements.

**Object:** An object is an instance of a class. It has a well-defined boundary of states and behaviours.

**Inheritance:** As with other object-oriented programming languages, the mechanism of inheritance is used by child objects to inherit the properties of their parent objects.

**Dependency:** A relationship by which an element or a set of elements (B) need other elements or sets of elements (A). If B acquires the specific characters or properties A possesses by association, the relationship is referred to as a Dependency. B, called the client elements, are “semantically or structurally” conditioned on how A, the supplier elements, are semantically or structurally defined [54:42]. Figure 2.6 shows an example of a dependency relationship.

The “Store Electrical Energy” and “Convert Sunlight to Electrical” use cases (UCs)<sup>19</sup> depend on the “Provide Electrical Energy” in Figure 2.6 above. The dependency relationship here is “include”. This means that the Provide Electrical Energy will include these other two sources of energy in its mode of energy transfer.

**Stereotype:** A Stereotype is a special type of metaclass, and is limited to its dependency on the metaclass. It has a dependency relationship with the metaclass that it extends in that, it is an extension of that metaclass. It is usable only in conjunction with the metaclass which it extends. A particular Stereotype can act as an extension to one or more metaclasses via association [54:258].

**Abstraction:** This could be considered as a dependency relationship that exists between two “NamedElements” or between a diverse sets of “NamedElements” that are representative of the same concepts of the whole, or of a part of the system(s), or of the system from a different perspective [54:38]. It is a dependency relationship that relates two or more sets of named elements which represent the same concept, but from different points of view. The relationship could be formal or informal, unidirectional or bidirectional, which is dependent on the stereotype, as well as those of standard specified stereotypes like **derive**, **refine**, **trace**, etc [54:38].

**Encapsulation:** They refer to the concept of the concealing of the details of implementation of a subset of a system by the use of data binding, thereby separating the system into well-defined boundaries and data structures.

**Polymorphism:** This refers to the concept of the ability of items to exist in different forms. For example, Figure 2.6 shows four actors associated to the “Solar Power System”. The “Load” and “Sun” actors are defined as blocks, which was described above. These two actors, although they are blocks structurally, have been used as actors. The other two actors, “Supplier” and “User” are the traditional SysML actors who use the system and provide materials for the system development, respectively.

---

<sup>19</sup>Where UC is used in this work, it denotes a use case captured as a SysML artifact.

There are three diagram types in SysML. These diagrams abstract their relationships according to the object-oriented concepts described above.

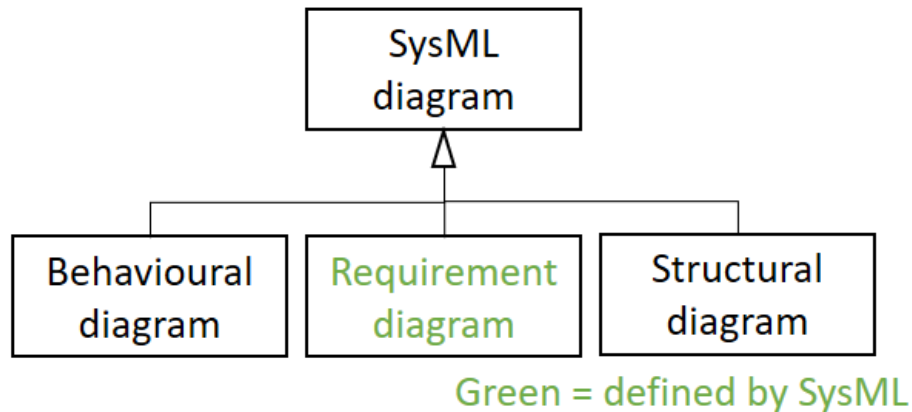


Figure 2.7: SysML hierarchy

The diagram types are 1) behavioural diagrams, 2) structural diagrams and 3) requirement diagrams, as shown in Figure 2.7. SysML also shows the system from different abstractions and views. Below is a description of the available diagrams in SysML and what they represent in terms of systems' abstractions and views.

### Requirement diagram

This diagram type is also defined by the SysML specification. They are used to visually relate the requirement artifacts to the model artifacts. The requirements are text-based and can be defined in the same area as the model, or in a remote area. Expressing the relationship between model artifact and the text artifacts (the requirements) that the model artifact fulfils are the common feature of this diagram type [55:186].

#### 2.5.1 Behavioural diagrams

As already described, SysML shows system views from different perspectives, and one of the three broadest perspectives is the behavioural view of a system. As the name suggests, the SysML behavioural view is used to represent the behaviour of the system as a model. They represent the response of a system to events and inputs into the system that changes with time, i.e. the dynamic behaviours of the system. These diagrams are shown in Figure 2.8 and are described below:

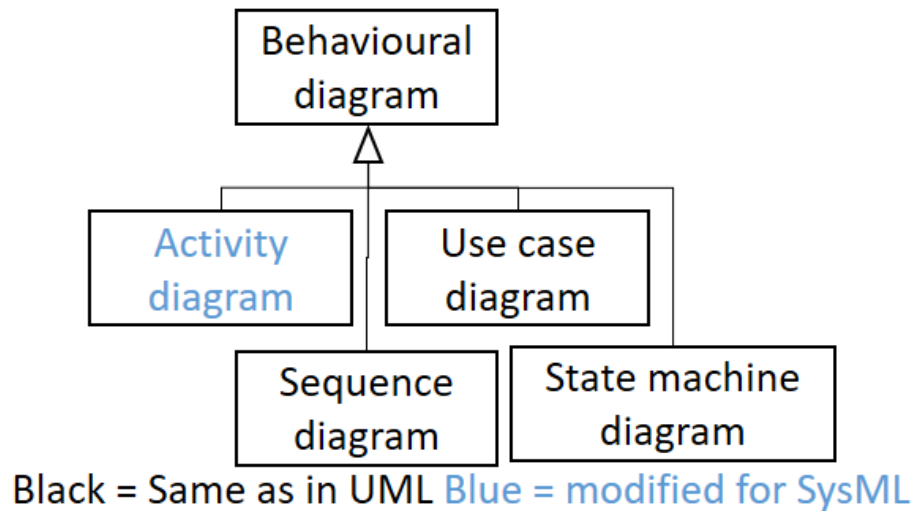


Figure 2.8: SysML behavioural diagram

**UC diagram:** A SysML UC is a representation of the relationship between the system and its surroundings - the environment that affects the system, including the persons that operate the system. These are referred to as actors. They capture what a system should do with respect to its environment (functional requirements). UCs can also be a nest of other UCs. Since it is a representation of the system and its relationship with its surroundings, the surroundings could be another system or subsystem. A use case diagram is a diagrammatic representation of these relationships and the actors. Three concepts are specified in a UC: the functional requirement of the system (action words, which denote the UC); the actor (the individual interacting with the system); and the relationship between them (represented as a line connecting the actor to the system).

**Activity diagram:** An activity in SysML is an extension of the UML 2 activity and represents the basic unit of behaviour. Using the activity diagrams, the flow and control of the input and output actions, as well as other actions that affect the system behaviour can be depicted [55:127]. One of the major differences between activities, as defined in UML and SysML respectively is, while control in UML activities allow actions to start, in SysML, activities can stop actions already executing.

**State machine diagram:** State machines define the “discrete event-driven behaviours” of a system model as a formalism of the system [54:303]. They describe the behaviour of a part or whole of system at specific time stamps. They also describe or express the behaviour of a system’s interaction sequences (protocols) of the system parts [54:303]. It is noteworthy that, the behaviour of a system is determined by a combination of inputs directly to the system and the previous state of the system. This can be expressed using state machine diagrams.



**Sequence diagram:** A sequence diagram expresses communication of events and occurrences in the sequence that they occur. It characterises the behaviour of a system as a sequential order of events that lead to other events.

## 2.5.2 Structural diagrams

The behavioural view of the modelled system depicts the dynamic behaviour of the system with respect to its surroundings. In direct contrast to the behavioural view is the structural view. The structural view captures the static structure of the system, in part or whole, and the static relationships of different parts of the system. They are described as those group of objects depicting a system’s static structure that are independent of time. They therefore, describe the architecture of systems. Elements of a system that remain the same at all times have attributes described as structural. They do not show any of the dynamic details of the system. However, the static structure may own dynamic structures or be related to them [54:683].

Figure 2.9 shows the SysML structural diagrams. There are four diagram types classified under the SysML structural diagrams. The “Structure Diagram” consists of “Block definition” and “Internal block” diagrams as modified concepts of the UML specification, as shown in Figure 2.9. The “Parametric diagram” type is a subset of the “Internal block diagram” type, as depicted in Figure 2.9. The following are brief descriptions of the diagram types available in SysML. Only the “Package diagram” is taken as-is from UML.

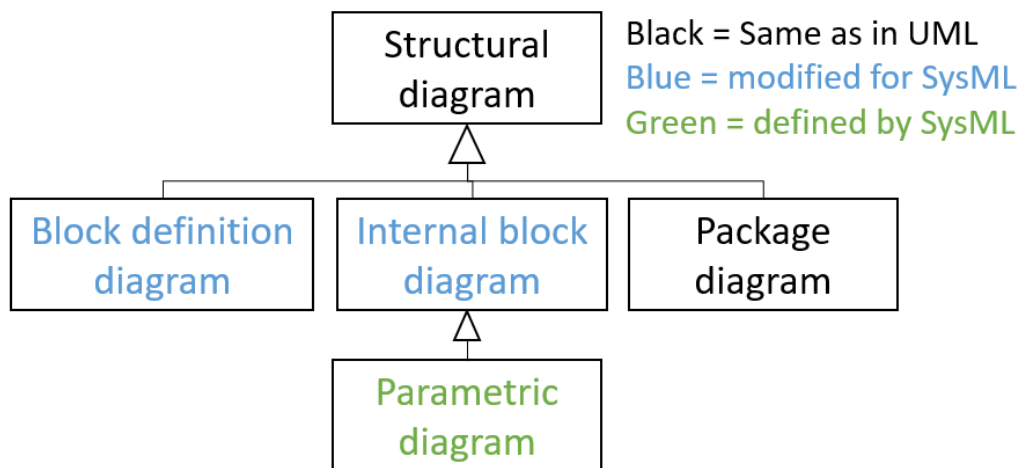


Figure 2.9: SysML structural diagram

**Block definition diagram:** Block definition diagrams display the relationships between elements in a hierarchically structured manner, as well as the value types of those structures.

**Internal block diagram:** This type of diagrams display the internal structure of a single block. The internal block diagram is used to display the relationship and constraints between internal components of a block [55:43].

**Parametric diagram:** The parametric diagram is a new concept defined by the SysML. This type of diagram is similar to the ibd, but with constraints upon each component. It is used to express the components in relation to their constrained properties and parameters. In parametric diagrams, only the constraints themselves and the parameters that are bound or associated to them are displayed [55:121].

**Package diagram:** A SysML package could be defined as a namespace for items that belong to the package, which includes the elements it is associated with. The elements are either owned or contained, as well as the elements that are imported into the package [54:239]. Packages can belong to another package (through nesting, or by association: merging or importing). The capability of SysML, as a formalised language, to organise and abstract systems as a composite of behaviours and structures, is a direct consequence of the capacity and structure that is provided in packages [54:12]. A package diagram is used to express packages and the items they contain or are associated with in an organised manner. It helps to group items together as belonging to one category.

Figure 2.6 is an example of a UC diagram. It is taken from a sample model for the design of a “Solar Power System”. The diagram shows six UCs: “Provide Electrical Energy”, “Store Electrical Energy”, “Convert Sunlight to Electrical”, “Report System Status”, “Install System” and “Configure System”. These UCs are defined within the boundary of the Solar Power System. That means, each of these six UCs are either used by, or use the Solar Power System. It also shows four items outside the boundary of the UC, but that directly influence the entire system. Two of them: “Load” and “Sun” are “Actors” in the form of “Blocks”, which are themselves, smaller systems, while the other two: “User” and “Supplier” are actors that have a manual part to play in the building and use of the system respectively.

There are two further points to note, with respect to SysML:

**Point 1:**

An operation is a behavioural feature, but it may be owned by static, structural objects, like an interface, a data type, or a block [54:114]. The “Load” UC in Figure 2.6 can be used to illustrate this concept. While UC itself is a block, which has structural attributes, the value of the electrical energy it “stores” is variable.

**Point 2:**

Similar to UML, SysML profile provide a mechanism for extending the capability of an implementation of the language. As SE domains differ, one domain might need

artifacts that are peculiar to its development of models. Profiles can be customised to provide those artifacts as a standard component of the specific domain, which can then be used by all who have access to the profile.

### **2.6 Summary**

In this chapter, ASIC was introduced as a composite system involving Hw and Sw parts. RE was also described as a process, outlining the process as comprising of inputs, activities and outputs. A dissection of the concept of requirements also took place in this chapter, thereby providing some contexts for the research objectives. The relationship between languages and RE process was highlighted, as well as their integration into SE. The chapter closed with a transition from naturally evolved human spoken and written languages to model-standardised languages.

## 3 State of the art

In this chapter, an overview of systems development from the viewpoint of MBSE that was discussed in the previous chapter is presented. The chapter starts with a presentation of the foundation of this thesis as an off-shoot of an on-going research. It continues with a presentation of some standards and technologies that emerged from the needs of related domains and complexities as that faced in ASIC development. The chapter concludes with a mention of evolution of possible solutions.

---

Management of requirements has been, and continues to be, a topic of discussion across all domains of systems development. The central focus can be expressed as a translation into “verification and validation”. Verification asks: “are we building the system right?”, while validation reasons: “are we building the right system?” [56:37]. Scholars continue to propose, and organisations continuously adapt their development processes to ensure as minimal a deviation from the intended product as much as possible. However, even the conceived products sometimes have deviations that are not covered by tests.

SE disciplines have grappled with the challenges posed by requirements handling. Processes, systems and technologies have emerged with a view to tackling these challenges. Some of the technologies relevant to the solution that is conceived for this are reviewed. On this premise, the abstraction of ASIC SE as a development workflow to evolve into a MBSE is discussed below.

### 3.1 Origin of this thesis

As mentioned in Section 1.1, NLS make up the bulk of requirements documentation. The customer requirements for ASICs are documented in NLS, and so are most of the FSs. Few of the FS parts rely on graphics. One part of enhancing the development workflow of ASIC is by MBSE, using the application of models to the workflow (see Section 2.3).

The foundation for this thesis rests on a currently on-going doctoral research that seeks an early formalisation of FS of ASIC SoCs. The research focuses on two main points, viz: 1) using models for the specification of SoC; 2) a model-driven approach to the generation of SoC. It focuses on representing requirements as models, rather than texts, because text-based requirements require the use of NLS. Figure 3.1<sup>20</sup>

---

<sup>20</sup>This diagram is adapted from the contents of a personal email

depicts the scope of the doctoral research work.

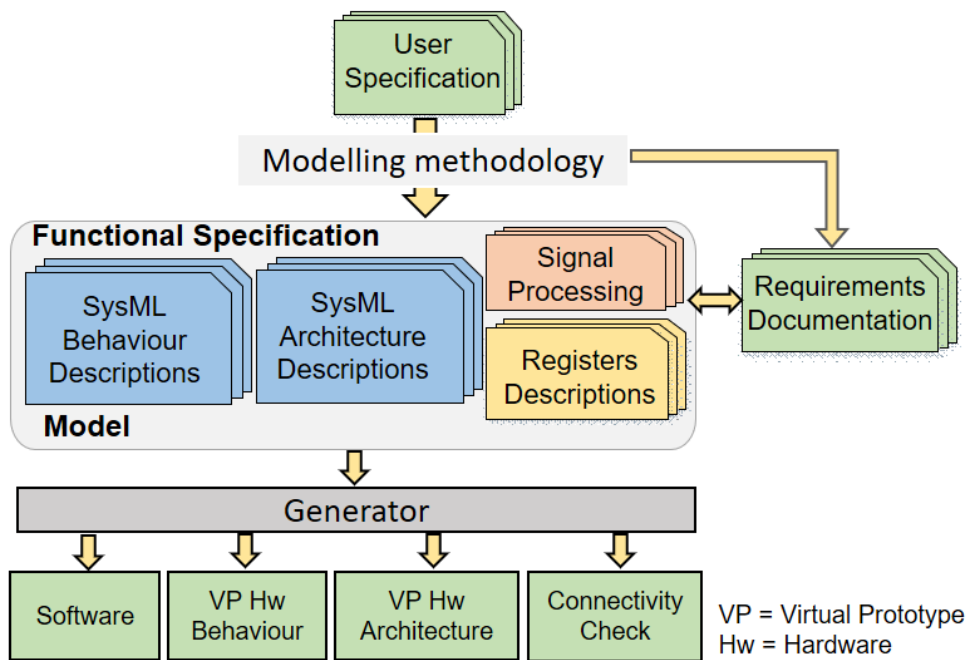


Figure 3.1: Workflow for a model-based ASIC development [57]

As shown in Figure 3.1, the “Architecture-” and “Behaviour Descriptions” are captured in the “Functional Specification” as model artifacts. These descriptions are modelable parts derived from the “User Specifications”. Figure 3.1 also shows that the properties defined for the “Registers” and the “Signal Processing” are also parts of the FS, equally derived from the CR. They are also models, but based on systems and tools other than SysML. As already mentioned, not all systems can be effectively described using SysML (see Section 2.3.3).

“Software”, “VP Hw Architecture” and “Behaviour” are then generated from the architecture, behaviour and register descriptions, as well as the signal processing models. The “Connectivity Check” of Figure 3.1 is used to test whether the internal connections defined amongst the different parts are fulfilled by the models and in the generated parts. These are typically generated as verification artifacts to assert that each of the generated components fulfil the connections they are intended to make to others. They are used to ensure that the designed relationships are maintained.

This thesis is conceived against the backdrop of the research work described above. Its scope is to prescribe a set of methods that will ensure verifiable traceability of SysML model-based artifacts to their text-based requirements. It is designed as an interface between model- and text-based documentation of the FS of ASIC SoCs as a precursor to combining both domains of documentation. That is, while in principle, specifications documented as models should not be repeated as texts, the focus at

this point is to establish a relationship between models and texts into “Requirements Documentation” as shown in Figure 3.1.

## 3.2 Reliance on natural languages

The ambiguity associated with NLs as an instrument and medium for requirements communication is one of the major challenges encountered in RE. Many different approaches to managing these challenges have been proposed in different domains. NLs are contextual, and without its context, a requirement is difficult to get right in its implementation. This has been considered and evaluated in research works like that of Karlsson et al [45:594] and Liebel et al [5], where direct interviews with the diverse group involved in a developmental project reveal gaps in communication. The difficulty of developing the user requirements into realisable functional requirements that can be translated to products was also reviewed. This results in challenges in developmental processes, due to the different contexts and roles each person in a diverse team has with the handling of the requirements. This means, different persons in a developmental process considers the requirements differently, and at a different abstraction level. The misunderstanding that characterises the transfer of requirements is illustrated in Figure 3.2 below.

Figure 3.2 is a ten-image storyboard that depicts the miscommunication in the development of a “swing” as a software project. The first image captures the swing as described by the customer, while the second image captures the project leader’s understanding of it. While both the customer and project leader understand the concept of a swing, their perspectives of this particular one quite differ. Those are also different from the perspective of the analyst, who analysed the requirements and allocated them to the programmers for development. The programmers developed a “flawed” system, and the business consultant describe it to other potential clients as the fifth image captured in the series.

The picture of the swing painted by the consultant is unlike what is in development. Although based on the same principles, they all have disimilar perspectives. Then, due to inadequate tooling, the project lacks the proper documentation. The billing and the support received for the project were disproportionate to what the project needed, however. The tenth image of Figure 3.2 shows the swing the customer actually had in mind. A series of misrepresentations of the customer’s needs produced the fourth image. The customer though, could have presented a picture, the tenth image, which might have helped align the perspectives of the other stakeholders to his view of the swing. Following the requirements from the first image of Figure 3.2 to the last reveals a dispersion of sort, depending on the stakeholder’s views of the system.

From Section 2.3, it is observed that NLs play major roles in a RE process. Managing these roles to enable efficient SE is consequently important in development projects.

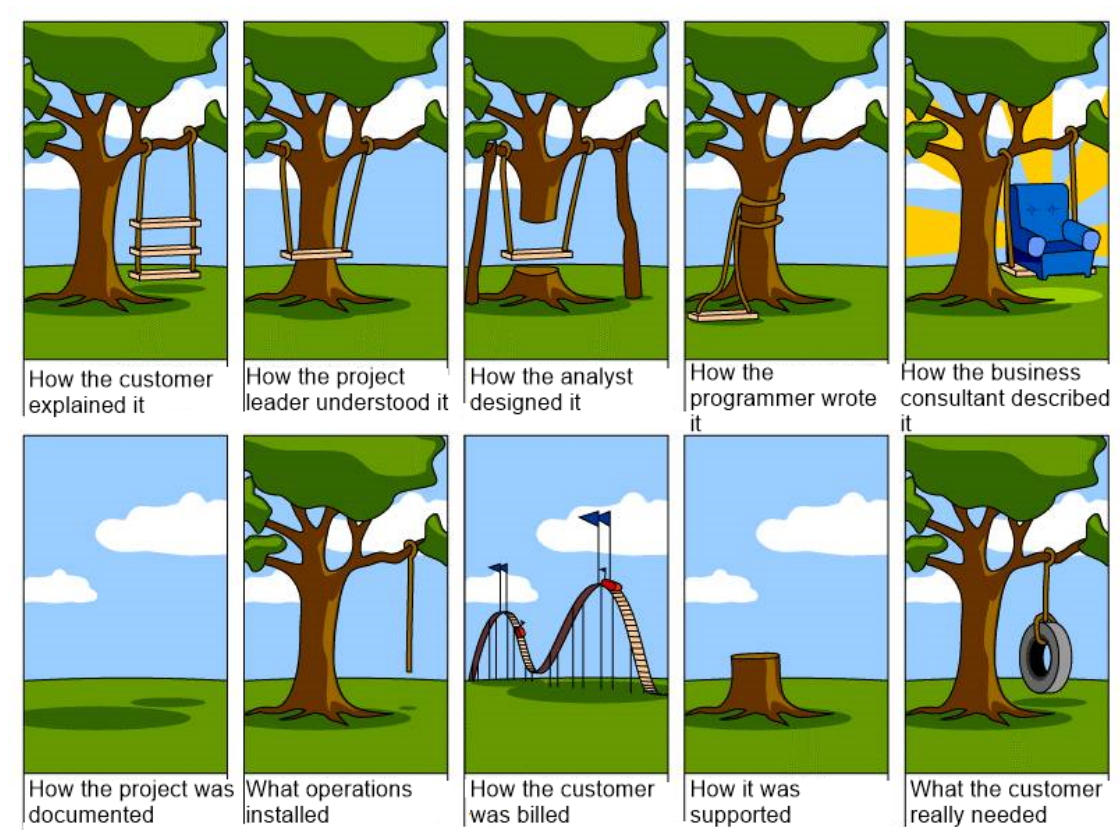


Figure 3.2: Classic requirement miscommunication example in system development [58]

On their part, Mich et al [2] evaluated the use of NL-processing (NLP) tools in RE. The focus was on using a computer-aided software engineering (CASE) system as a NLP tool to aid software developers and engineers in processing requirements from the early phases of development, namely requirement elicitation. Their approach considers that the use of NLP tools produces three types of data: 1) syntactically, semantically and/or pragmatically analysed data, 2) data in text formats, in either a natural or artificial language, and 3) summaries or templates of the input data in a structure different from what was given to the system.

### 3.3 Document-based systems engineering

Closely related to, and an integral part of, the problems inherent in NLPs are the difficulties of managing systems development activities that are dependent on the physical transfer of documents throughout the lifetime of the project. This is known as the traditional approach to SE and is referred to as the practice of DBSE.

The requirements and model items that are produced in a SE developmental project are called artifacts. These artifacts are used to convey and document information

related to the project as described under RE process in Section 2.2. Generally, artifacts like “*requirements traceability and verification matrices (RTVMs)*”, *architecture description documents (ADD)*”, etc. [53:3] are generated during the life time of a project. When the project is based on DBSE, these artifacts are developed manually. Projects that are based on this approach generally rely heavily on spreadsheets, word processors, presentation tools, cardboard sheets, etc. These documents are written by different people and teams.

In theory, any changes to any item in any of the documents must be followed by a notification to each member or team affected. The notified stakeholder has to effect the communicated changes where applicable. In practice, however, it is often not realisable. Consider a “name change” to an item in the trail of documents shown in Figure 3.3 as an example. After elicitation as described in Section 2.2.1, a name of an artifact is changed in the user requirement specification. This changed name has to be traced to every part of each of the documents it appears in, and the changes effected, each time there is a name change.

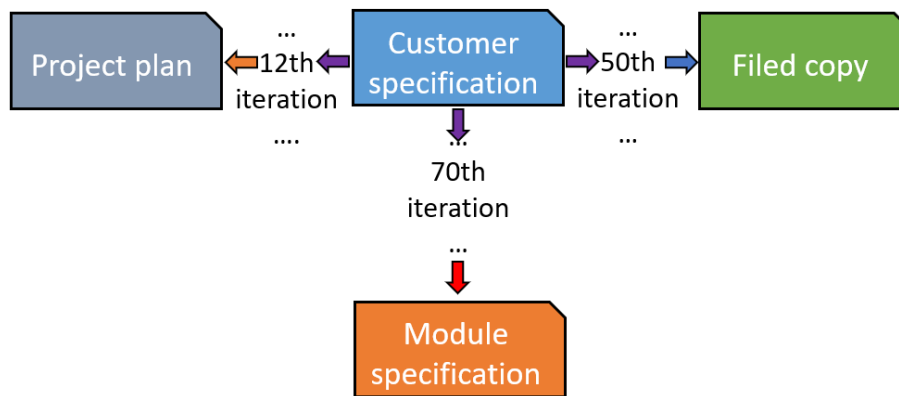


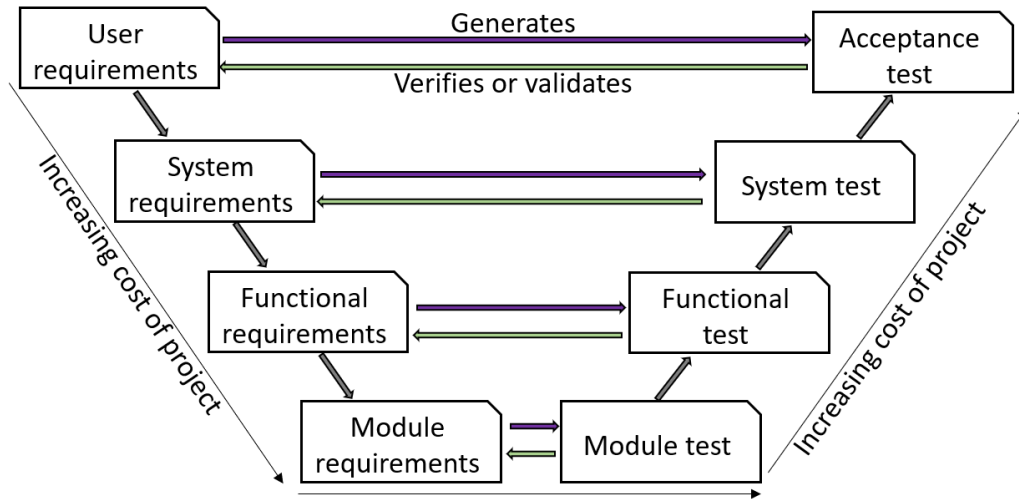
Figure 3.3: Requirements documentation name mismatch

Figure 3.4 is a V-Model. The left-hand-side (LHS) represents the design phase, while the right-hand-side (RHS) represents the integration and testing phase of a project development. The closer a change of requirement is to the upper part of the design phase of Figure 3.4, the lesser the cost incurred on the project development [59:8] [50:page 2.6 of 10]. Consequently, the closer to the topmost part of the integration and testing phase a recall of requirements is, the more expensive the project development. Test and report documents are manually produced and transferred for each stage of the project. The purple arrows in Figure 3.4(a) indicate the manual procedures to produce the RTVMs and ADD.

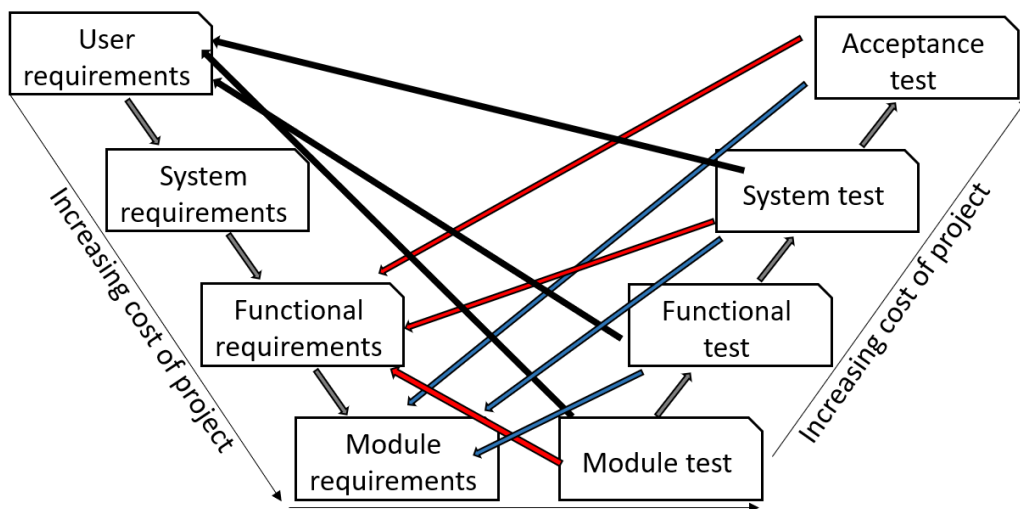
Artifacts needed in the integration and testing phases (RHS of Figure 3.4) are produced at their corresponding design phases (LHS of Figure 3.4). In ideal cases, this manual transfer and usage of the produced artifact is expectedly sufficient. If the test results are unsatisfactory, requirements are traced to the corresponding phase



on the LHS and are reviewed, as represented by the green arrows in Figure 3.4(a). However, due to cycles of modifications that often occur in development as already illustrated with Figure 3.3, there are often inconsistencies. Due to these inconsistencies associated with manually editing and modifying the requirements, tracing of the requirements to their sources of deviation causes the trace to be scattered all over the entire LHS of Figure 3.4(b) as represented by the red, blue and black arrows.



(a) Ideal case



(b) Obtainable case

Figure 3.4: Verification and validation with DBSE

Keeping the names (and other relevant attributes and properties) of artifacts consistent is, therefore, impracticable in the practice of DBSE. The desired consistency

of requirements cannot be met in a production environment using this approach to SE. This is especially the case for projects with a lifetime of ten to fifteen years, as is the case of ASIC development. The DBSE approach to SE is cumbersome, expensive, error-prone and inefficient [53:3].

## 3.4 Model-based systems engineering

As already discussed, the use of NLS is problematic, since it is too often open to interpretation. Furthermore, DBSE is expensive and inefficient, as deduced from Section 3.3 above. To reduce the problems that characterise NLS and DBSE or eliminate them where possible, MBSE was conceived. MBSE seeks the evolution of a system of best practices that systems engineers can adopt for a more efficient analysis and management of risks involved in development.

To understand the concept of MBSE as a SE approach, a proper definition is reviewed. MBSE is defined in [60] as an established framework for the conceptualisation, design, development and manufacturing processes. It is focused on applications and expression of precise architecture frameworks, design guidelines and models for the activities towards the realisation of an engineered product. It also continues after the product is realised, because it is also concerned with the maintenance of the product, and throughout the product's entire System Development Life Cycle (SDLC). The INCOSE defines MBSE as

*“the formalized application of modeling to support system requirements, design, analysis, verification and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle phases. MBSE is part of a long-term trend toward model-centric approaches adopted by other engineering disciplines, including mechanical, electrical and software. In particular, MBSE is expected to replace the document-centric approach that has been practiced by systems engineers in the past and to influence the future practice of systems engineering by being fully integrated into the definition of systems engineering processes”.* [61]

A **model** could be described as a representation of the concept, environment, phenomenon and structure of a system as an abstraction of its reality. The representation could be graphical, mathematical or physical, with respect to the system's environment. It is called a model when just enough of the system components necessary to represent it in parts or whole, in a functional state, is used to describe the system [59:3] [62]. Modelling of systems as a composite representation at different levels of abstraction help to separate the systems being built into their areas and domains of concern<sup>21</sup>. Both the requirements the models (or the specific part of a model) that fulfils them can be represented. The abstractions in models help

---

<sup>21</sup>“Areas and domains of concern” refers to “separation of concern” described in Section 3.5

to view a system as a combination of components, processes, tools, environments, and other variables required to realise the system. It also shows the interactions and communication between the different abstractions of the system. This helps to promote the integration of systems and processes, including the enabling systems and communication mechanisms that fulfil the systems' requirements.

MBSE, therefore, implies the use of models in the design and development of systems where had hitherto been based on the document approach highlighted above in Section 3.3. "A picture", as the saying goes, "is worth (more than) a thousand words". One of the ways MBSE tries to eliminate the ambiguity in development is to define diagrammatic artifacts that convey meanings to systems engineers universally. It could be for a specific system domain or across multiple domains. These diagrammatic artifacts are understood in the same contexts by the systems engineers.

In an attempt to derive a fitting definition for MBSE, Barnhart [63] observed by deduction, that there are two parts to defining what MBSE is: the first being that, it is a process whereby visual representations for a system being built are generally formalised; and the second being that these visual representations also provide a means to capture specific important "information about the systems" to enable a successful realisation of such systems. This means that, with the help of visual and contextual diagrammatic representations and the captured system-specific information in its environments, the system can be uniformly and universally described.

There are different levels or views of a system. These levels are contextually regarded as system abstractions. A system may be composed of sub-components, other systems or subsystems. Looking at the whole system is viewing the system at one abstraction level. Viewing the system as a combination of two or more systems is another abstraction level. When the system is considered as composed of components or units, those are further abstraction levels.

The formalisation of models simplifies the concepts of the system structurally, and its relationship - mathematical or physical - with its surroundings. It abstracts the system in such a way that the behaviour, meaning and realisation of the system is clear to the analyst and the developer, even before a first prototype of the system is produced. Using MBSE, the interactions of the stakeholders and at what stage of the developmental process are also captured. In a nutshell, the complexity of a system - from its conception to realisation and down to its retirement, including the activities, roles and individuals or teams needed can be represented as formalised models. The form of communication between the collaborators stakeholders and the actors can also be represented in formal models. This concept of abstraction has precipitated a system of separation of concerns which are discussed below.

## 3.5 Application lifecycle management

Managing a development project that involves several departments, people with different roles, and with upwards of a thousand requirements - as is the case with ASIC development - is challenging. It potentially involves managing different RE processes. To better manage such developmental processes throughout a product's entire life time, a process known as Application Lifecycle<sup>22</sup> Management (ALM) evolved. This management starts from the product's conception phase until retirement. ALM is software engineering domain's adaptation of project management in contemporary developmental process known as Production Lifecycle Management (PLM). It is defined as

*“[...] the overall business process that governs a product or service from its inception to the end of its life in order to achieve the best possible value for the business of the enterprise and its customers and partners”.*

[64]

This incorporates all the measures and activities defined in Section 2.2. On their part, Tüzün et al [12] considers it to be a “recent paradigm”, with the same purpose as what was defined for SE by the INCOSE (see Sections 2.3 & 3.4), but in the area of software development. It was considered as the integration of governing procedures for software systems' development. Now, ALM considers development of a system as comprising of both software, hardware and other non-functional parts, like cost of development and planning. There is also the Software Development Life-cycle (SDLC), which is another nomenclature that is used specifically for software development activities. SDLC, like the other developmental processes, considers phases and steps of development, including planning, designing and building of the product. The central purpose of ALM, PLM and SDLC alike is to reduce cost of production, while improving the quality of products through the use of standard and repeatable measures.

Scholars have debated whether ALM is a methodology and technological framework [65], a paradigm [12], a system [13:3] or a process [64]<sup>23</sup>. Clearly, ALM helps break a system development process into tangible separation of concerns. A “separation of concern” could be described as, making a system visibly appear as distinct units that work together to achieve a common goal. It also highlights whose responsibility it is to provide the individual units of the system. The role of providing the units might be the responsibility of a distinct person or actor. It also helps to integrate these separate “concerns” into a functional and practicable development chain. This also means, a new or substitute entity can easily take over a role that was previously assigned to another entity.

---

<sup>22</sup>It is written as Life-cycle by some

<sup>23</sup>This work considers ALM to be a methodology, as well as a technology. The clarity is inferred in its contextual use

## 3.6 Change and configuration management (CCM)

As already mentioned in Section 2.2.1, requirements are susceptible to change. Changes without preparedness hinders a development process. These changes could jeopardise an entire developmental process, if not properly managed or planned for in advance before they occur [53:85]. As changes can occur at a moment's notice, it is important to have planned for alternatives. Planning for eventualities requires the use of adequate tools for documentation of activities and schedules during the course of development. Change and Configuration Management (CCM) systems are ALM-based tools used to plan and manage changes in a project. CCM in software development comprises of two parts of a developmental process that are often considered together. In the context of this work, however, the two are, first and foremost, considered distinct from each other, as “Change Management” and “Configuration Management”.

### 3.6.1 Change management

Changing requirements for a system could, sometimes, mean a re-work of an entire process. Due to the iterative nature of requirements processing discussed in Section 2.2, a system development that seemed feasible might eventually be discontinued, if certain stakeholders reject a change in requirement. Other requirements could also change; personnel changes do occur in a development process. In the event of personnel changes, prior discussions had in the development project are not automatically transferred to the replacement personnel. Trainings help develop people's knowledge of a project domain. Training sessions that were had and the resultant know-how do not get transferred automatically to the new persons, when changes to personnel happen. These personnel changes also occur in manners that replacements are not readily available. When such changes occur, contexts of requirements might go missing, because they stayed with the former persons.

Requirements are also not just products' requirements. To ensure a product meets the requirements set out at the start of the system design phase, the system is tested and validated against the agreed requirements. There are testing criteria to observe. These could refer to other tools and equipments that must be available for the tests to be carried out. They may also be related to licences and licencing. They could also refer to other entities that are necessary for development, verification and validation activities to be performed. According to [66], the Project Management Body of Knowledge (PMBOK) designates change management as those changes related to baselines, plans, as well as processes of a project. This category deals with changes that affect the product indirectly. As an illustration, to fully understand the concept, if personnel changes are made or to be made, requests are placed for such changes to be implemented. It is, therefore, important to note all these changes; who requested for them; why they are requested for; and by whom they are implemented. In other words, to modify the process, plans, or baselines, a record has to be kept in the form

of activity logs.

#### 3.6.2 Configuration management

Traditionally, in Sw development environments, change management was considered as the defined procedures of making changes to codes. This has been replaced by Configuration Management, which is now used to refer to the actual changes in plans and codes that have direct impact on the products.

Sometimes, the changes required in the project area are operational changes. Newer methodologies have described a process called “Development and Operations” (DevOps) that capture a lot of these changes. In DevOps, continuous integration (CI), continuous testing (CT) and continuous deployment (CD)<sup>24</sup> necessitates the delivery of minor developments in parts of a production system incrementally. Oftentimes, because multiple people work on different parts of the system concurrently, or at different times, it is necessary to integrate new developments into the system in bits. The system is tested under controlled environment to ensure its stability with the updates added, before the system is released as a newer version (stable release). This often requires the use of a central server as a code repository. Typical examples of such systems are the use of Git and GitHub as code repositories.

In the case of changes to requirements in ASIC development, the modifications directly impact the ASIC. This modification of the requirement is actually to the requirement management system, and not directly on the ASIC. This means an overlapping of management domains. As mentioned earlier, tools with the capability to monitor, create and log change requests and activities, and report on these activities to all concerned shareholder is increasingly desired.

#### 3.6.3 Source control management and version control

CCM is really a feature of Source Control Management (SCM)<sup>25</sup>. SCM is a system of control and management of changes to a system that is in development. It is mostly applied in DevOps. The source of updates to a system, especially in a distributed team, is monitored. As mentioned in Section 3.6.2, the updates to the system is first integrated into the system in a test environment, and checked for errors, before releasing the integrated system as a newer version.

Repositories (e.g. Git) are used to maintain system codes to which several people could have access, and deliver their codes to, in order to add new features to the system, after a set of functional parts of system is completed. These codes are then

---

<sup>24</sup>Continuous Delivery is another variation of the use of CD in DevOps parlance.

<sup>25</sup>SCM has evolved from “Software Configuration Management”, where a purely software system is being built, to include entire system of systems. Some also refer SCM as Source Configuration Management.

integrated into the system, and tested to ensure a successful integration. If the integration fails during the testing phase, the system is reversed to a last known working configuration (version). For this to be implemented, the process has to be versionable. Being versionable means the SCM system must have a mechanism for setting baselines of the system. Versioning a system could be referred to as reaching a milestone in the system or project development, and marking it as a benchmark on which further development is based. This is known as “Version Control” (VC). The method of release of major and minor versions of a product is an example of VC implementation. For instance, a product might have major release versions of 1.0 and 2.0, and versions 1.x and 2.x as minor releases. In the same way, a model can be baselined and versioned as benchmarks in a model-based development environment.

## 3.7 Quality management (QM)

While CCM is used to request and make changes to tasks and requirements, a separate ALM-based tool is used to plan and execute tests on the implemented codes. These plans are to be carried out at different stages of the development according to the process model in practice, e.g. V-Model (as discussed in Section 3.3). Tools that are used for such activities are called Quality Management (QM) systems. QM systems provide the platform on which test plans are designed, created and managed. Tools in this category are often also versionable. Running of tests from QM tool is done with the help of tests environments provided by the respective QM tools. These environments are planned, generated and managed depending on the use case. With capabilities delivered through a combination of CCM and the QM systems, software product are often delivered faster. The visibility of the status of the product with respect to its delivery of service can be seen and evaluated in real-time. It also aids teams and cross-teams in collaboration, because collaborators are informed of the progress of the development work of the team. In essence, using CCM, deliverables of sytems are modified and in real-time, those changes are implemented and tested using the QM. This helps to determine the suitability and compatibility of the newly developed part of the system with the whole of the system in real time.

## 3.8 Open services for lifecycle collaboration

Formerly, it was difficult to integrate sets of tools as a tool-chain in a software production environment, especially when the tools are from different vendors. To use a platform with tools from different vendors would require a vendor-based tool-chain [67] [68]. In other words, using tools provided by different vendors for collaboration was impossible. This incompatibility of vendor-specific systems in a PLM sample space is being replaced by a much more robust development space with the introduction of the Open Services for Lifecycle Collaboration (OSLC) standards.

The OSLC defines Resource Description Framework (RDF)-based specifications for the use of Representational State Transfer (REST) Application Program(ing) Interface (API) in designing and developing of software systems and tools [69:155] [68]. This specification uses the “Create, Read, Update and Delete” (CRUD) operation format to ensure that a client of a service with the right authorisation can CRUD resources as needed. This will depend on the nature of the resource and the client or server.

The OSLC standard also defines the use of HyperText Markup Language (HTML)-based standards, to enable the integration of tools from different vendors in a production environment, using “linked data” methods. The relationship between the sender and receiver is on client-server basis. The server specifies what “shapes” of requests it accepts, and the shapes of data it sends back. Any request to it must contain those shapes to receive definite responses for each request data. A client that wants information from the server will have to present its web-based requests in the shapes specified by the server [69:155].

Both standards enable OSLC-compliant tool-integration. The use of standardised REST API is beneficial to tools vendors, tool buyers, and tool users alike. Figure 3.5 depicts OSLC as helping the integration of heterogeneous systems. In Figure 3.5(a), four different servers have to be provided to service systems that are based on four different data types or protocols. That was before OSLC. With the introduction of OSLC, services are much more independent of the server and client type. Figure 3.5(b) shows a client- and server-neutral system, with a common standard that enables communication and information sharing amongst them.

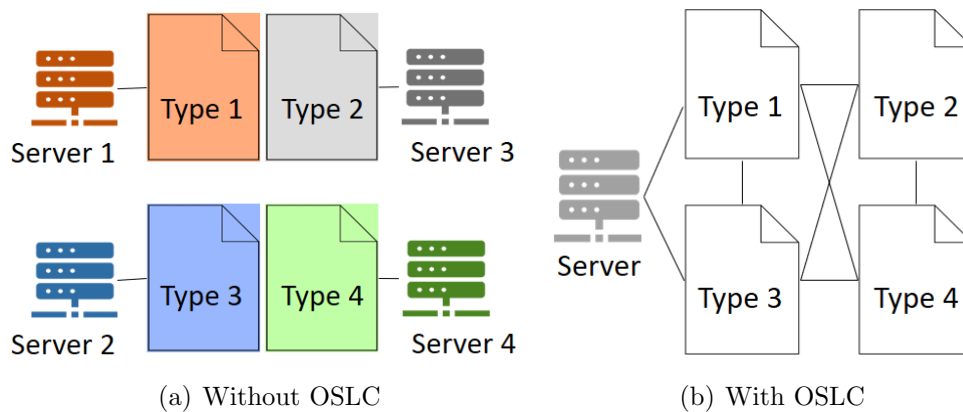


Figure 3.5: Exhibition of the use of OSLC in a server-client environment

APIs are the interfaces between different parts of a computer program. APIs provide just the sufficient layout or structure information of a program that a developer needs to implement a particular sequence of operations, depending on the domain.



The standard ways of expressing APIs are “as a set of operations”, definitions of associated data, and as “semantics of the operations on some underlying system” [70]. They are used to request and deliver information from source to destination according to specific formats or standards (specifications). APIs can be for a web application or web client (web API) - which usually means hypertext transfer protocol (HTTP), or for many other applications.

According to Emery [70], to be useful, APIs have to be expressed in, and bound to, the context of an executable computer program. To get an executable program requires the use of a programming language. It further states that, access to APIs are provided with the following two defined terms that characterise APIs solutions across domains:

**Interface:** This refers to the abstraction that characterises the requirements for the behaviour of the interface within which the API is contextualised.

**Binding:** This refers to the context of the programming language for which the API is to be realised, leading to two related problems: one of which relates to the mapping of features and capabilities of the specific language interfaces, while the other deals with the specification of the bindings of the specific language as a standard [70].

REST API is a style of architecture defined for “distributed hypermedia systems” [71] with the following six guiding constraints as described by Fielding [72] in his doctoral dissertation:

**Client-server reference:** a server and a client are separate entities that can evolve independently of each other. That is to say that, a server and the database contained there-in can be changed or modified, but would not impact the nature of the information that it sends to a client. Likewise can a client be changed or modified, and not change the nature of the information it requests from the server or that it communicates to other clients that share the resource. Separating the system in this manner ensures a client-server system to be scaled up without the need to adjust other systems or products within the system.

**Stateless behaviour:** following the server-client relationship already established, a call to and from a server (state) should be independent of a previous or future call (past or future state) of the client or server. Rather, each call to and from the server be treated as specific to the particular instance of the call that it satisfies. Being that calls are inherently made from clients to servers (i.e., servers do not initiate calls to clients, rather a system making a call to another is of that instance, a client), the state is stored at the client, and not at the server. In other words, the response a client receives from a server at any instant is independent of any previous or future calls from the client (or any other client) in a system, but of the particular instant it satisfies. The server (and hence, the communication system) is therefore, stateless.

**Cacheable resources and information:** despite the statelessness of REST API calls as described above, servers can inform clients in their responses, whether a particular response data is cacheable, or not. As established, servers do not cache the calls made by clients, but clients can cache the information they receive from servers. A server in its response to a client request, though, has to inform the client whether the information contained in a response would still be valid in the future; and if so, for how long. This information is then considered by the client, whether to save it, or not, for future reference, instead of making calls back to the server for the same information within the window the same information received previously still holds valid.

**Uniform interface across platforms:** to ensure that systems can be modified or changed without the need to change other systems that communicate with them in a platform, a common set of interface is required. This interface is independent of the underlying structure of the communicating systems. A common format for the communication of systems is standardised, so that each system is able to CRUD resources according to its allotted access or authorisation, to enable the communication across entire the ecosystem of the whole communication platform. There are three visual formats of a typical REST API resource: Javascript Object Notation (JSON), XML, and regular text file formats.

**Layered hierarchical system structure:** defined as a hierarchically structured system in a layered communication, whereby each layer has its unique and specific function in enabling communication and data sharing, ensuring that each layer communicates and shares data with the layer below and above it, without knowledge and irrespective of what structure or format of data communication is applicable in those layers above or below.

**Code on demand (optional) feature:** REST's clients' functionality is extendable when the codes are downloaded and executed as applets or scripts. By reducing the number of features that have to be pre-implemented before integrating into the system, the management of the clients is simplified [72:84]. Using "Code on Demand" capabilities, the applets or codes that is to be used for an application can be transmitted through the APIs. It enables the creation of what is referred to as "a smart application" which does not rely only on its own code structure to perform the required operations of the system [73]. It is an optional feature though, because of its drawback namely: visibility reduction.

The use of REST API in the OSLC specification guidelines guarantees the integration of several tools, building a tool-chain comprising of different vendor-specific systems and tools, as well as the ability to share data consumable by different systems. As described by "OASIS Open Projects"<sup>26</sup>, the benefits of OSLC include:

---

<sup>26</sup>Since May 2019, OSLC is an OASIS Open Project [68]

**Vendor neutrality:** OSLC standards prescribes vendor neutral implementation.

One application can be used for the generation of some stated data, and can be replaced with a different application that performs he same function, without the need to change any other systems dependent on the data being generated.

**Re-usability of data:** The data generated by a standard OSLC-conformant application are re-usable by other applications that are also compliant to the OSLC standards. Data generated by the one is consumable by the other application. In Figure 3.6, there are four applications used to demonstrate this re-usability. The shapes in each tool represents as they are specified by OSLC. The specific data needed by “Vendor A” might be available in the tool from “Vendor B”. Vendor A requests for the data, but uses it in a different format.

**Easy integration of applications:** With OSLC, it is easier to integrate one’s own existing or new applications, or to modify the data in the context of the new application. It is also possible to integrate data from different vendors or applications into an existing or new application, irrespective of the originating system or database. Due to the re-usability of data described above, it is easier to use two different applications for different purposes. The CCM tool may be developed by the one vendor, while the QM system is developed by another. If both tools are OSLC-compliant, data (represented in Figure 3.6) as shapes can be easily reused. This helps to create an integrated vendor-neutral tool-chain.

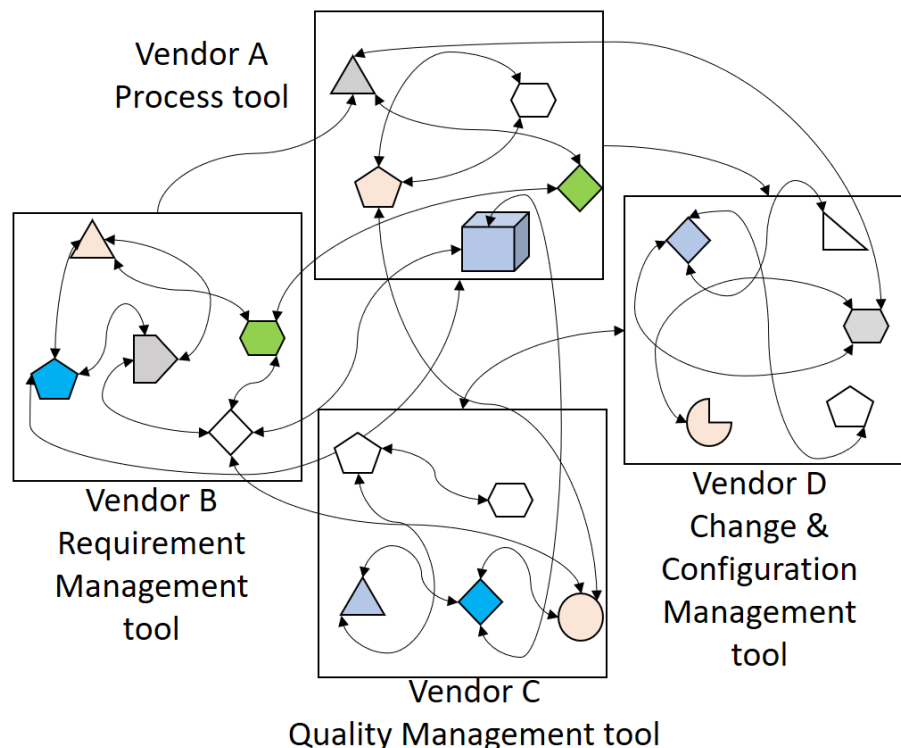


Figure 3.6: Integration of systems using OSLC

**Linking data together:** Due to its vendor-neutrality, OSLC standardised data is obtainable in a network of systems with diverse OSLC conformant applications, and the data generated by this network of systems is consumable by the respective applications.

**Visualisation and analyses of data:** OSLC standards make it easy to develop applications and methods to help visualise and analyse data in forms like graphs, tables, and trees. When tools from different vendors are to be integrated, the information contained in the data generated and transferred by each tool has to be visible to the integrator. Assuming tool C is to be integrated with the other tools, as shown in Figure 3.6. The data structure and types delivered from tool C has to be transparent, so that tools A, B and D can be configured or programmed to use the data produced by the tool C, if they depend on any such data.

**Ease and convenience of making decisions:** With these benefits stated above (and much more) delivered through the use of OSLC specifications and standards-conformant applications, it is easier for individuals and organisations to make business decisions based on the needs of their projects and businesses. Organisations can make decisions concerning what tools to use for change, configuration, requirements and quality managements. Other automation and monitoring tools can also be added to the tool-chain (as shown in Figure 3.6), according to business and project needs, if all are OSLC-compliant.

To make a successful REST call, a REST client provides information that the server has described as necessary for a service to be delivered to clients by the server. This means that, for any call made to the REST server, the call must be in a format defined, provided and serviced by the REST server, if communication between the two must be completed.

## 3.9 Traceability methodologies and types

UML and SysML are state of the art modelling language specifications and standards that promote MBSE. However, they do not proffer any methodologies on how to use them. They are neither systems nor tools of implementation. Systems and tools that are compliant to these standards can be developed. There are also other such languages and tools that are being used for the practice of MBSE. The Embedded electronic Architecture STudy - Architecture Description Language (EAST-ADL), for instance, is one of such languages used in the development of embedded electronics for automobiles. The use of these standards-based tools have to be deployed with a set of principles of use that is observed by collaborators for a development project, in order to achieve specific targets.

One of such principles is to link the requirements to the models that fulfils them. The tools of implementation usually provides such mechanism. The stakeholders

have to define how they are used, though. The work by Albinet et al [8] describes a methodology for tracing of requirements to models in embedded systems, using a combination of EAST-ADL, MARTE<sup>27</sup> and SysML. The traceability described relies on SysML for its capability for modelling of requirements. Traceability of requirements to their logical, functional and contextual roots, stems and branches is one of the focal points of MBSE. It, however, does not consider the abstraction level at which requirements traceability is to be integrated into models in the area of ASIC development.

While traceability of requirements is one the focal points of MBSE in general, and RE in particular, requirements reuse is another very important tenet of the model based approach. Re-working or re-specifying a set of requirements for similar systems belonging to two or more different series of ASIC is time wasting and unnecessary. Furthermore, using existing model artifacts for the development of a new set of system components is encouraged in the practice of MBSE, only making adjustments where necessary. A SE environment that has evolved into a “data-centric” practice, opposite to a “document-centric” one, is an ideal MBSE [59:7].

The INCOSE Object-Oriented Systems Engineering Method (OOSEM) is a tool and vendor-neutral methodology defined to support the practice of MBSE. Although it is independent of tools and vendors in its approach, what is publicly available about this methodology is its approach to “Requirements traceability”. Six steps are proposed in this approach. They are: 1) analysis of the stakeholder needs using use cases and enterprise models, 2) definition of the system’s requirements by using use cases, as well as elaboration of the context of the requirements and use cases, 3) defining a logical architecture for the system, 4) synthesising the physical architectures of the systems from the logical context, which include the data, hardware and software architectures, 5) evaluating and optimising alternative models and requirements as replacement use cases, 6) verification and validation of the systems according to the defined and analysed stakeholder requirements.

An IT managers’ opinion poll shows that thirteen percent (13%) of project failures is caused by incomplete requirements [74:1]. This was adjudged as the main reason for project failures. Incomplete requirements could be at any stage of the development phase. When a context is missing, and it is supplied and not communicated back, it goes into the next phase as an incomplete “point” in the development, whether it is discovered or not. The same poll opines that, 13% of projects succeed, because of clearly stated requirements. So, on the one hand, we have failure of significant magnitude, if it were to happen in semiconductor markets of the automotive industry, as the volume of production was projected at forty-six billion dollars as of 2019 [75:3]. And on the other hand, we have success rate that is about a 180-degree

---

<sup>27</sup>Modelling and Analysis of Real Time Embedded systems (MARTE) is also standardised by the OMG®.

turnaround of what a failure could be. These are typically software services, systems and/or “value-added” projects. However, considering that the use of software in embedded systems domain, particularly in the functionalities of ECUs in cars are expected to be dominated by software implementation (about 80%) [32:23], the appropriate handling of requirements with respect the level of abstraction required at each level, to keep its implementation in accordance with the context for which the requirement is desired and fulfilled in tact, increasingly become an area of focus to software teams.

Recently, methodologies and processes like Agile and Scrum that prescribe approaches to requirements handling have been proposed and adopted by organisations. These proposals continue to evolve to better solve challenges that arise with requirements and process complexities. In Section 1.1, NLS are said to be the most predominant form of requirements specification. Other forms of documentation are formal and structured language make up the other 21% of requirement documents [2]. That is, 79% of the communicated requirements is at risk of misinterpretation. This means that miscommunication during transfer of requirements is more likely.

## Requirements traceability

Among other things, this work is intended to implement measures that promote and ensure the traceability of requirements to their sources and destinations at all times. But first, requirements traceability as a concept has to be understood. The following source conceives that:

*“Requirements traceability refers to the ability to describe and follow the life of a requirement, in both a forwards and backwards direction”* [76:1].

For a system to be useful as planned, the requirements for the use cases has to be satisfied by the system after it is delivered. And as already discussed in Sections 3.3 and 3.6, requirements without proper management could lead to excessive cost of a project.

Based on the definition above, gathered requirements should be linked to every part of the system that it impacts. Every behaviour, function and structure of a system is based on pre-defined requirements. Each pre-determined use case and function should be verifiable by being traced to the requirement on which it is based. It is pertinent to say that, however well structured and detailed a requirement is, it is only as good as its traceability. A requirement, without the extensive relationship it requires, will not be properly verified. Its relationship to other artifacts is vital to its fulfilment and verification. The relationship of a requirement to a model is either direct, when they are linked directly by a dependency, or inferred, i.e. linked by an indirect dependency. This is illustrated with Figure 3.7.

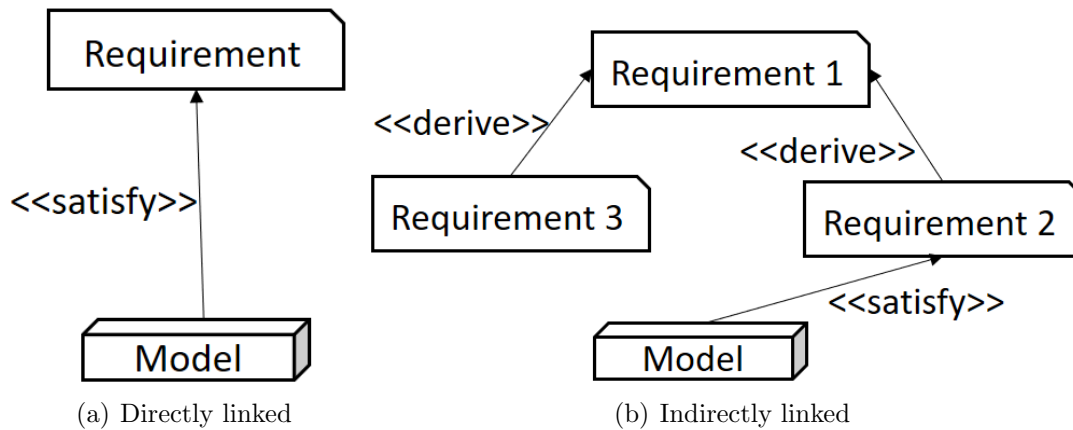


Figure 3.7: Associating models to requirements artifacts

In Figure 3.7(a), the model element directly satisfies the requirement. That is, they have a direct relationship. Requirements 2 and 3 are directly linked to “Requirement 1”, as seen in Figure 3.7(b). They both have a “derive” relationship to Requirement 1. This means, Requirement 1 is parent to both 2 and 3. However, only Requirement 2 has a model linked to it. It can be concluded, that Requirement 3 is yet to be fulfilled in the setup of Figure 3.7. Every model artifact must be linked to at least, one requirement, meaning it has to be in fulfilment of some requirement<sup>28</sup>. On the other hand, every requirement must be directly or indirectly linked to a model artifact. Meaning a requirement has to supply some information that is to be fulfilled. Figure 3.7(b) above illustrates that, the model directly satisfies Requirement 2, but also indirectly satisfies part of Requirement 1. The model has no relationship to Requirement 3. The figure shows, therefore, that there is still a missing link to Requirement 3, as it has no relationship to a model.

Generally, traceability of requirements is done with methodologies and tools that provide visual relationships between requirement items, either to other requirements, or to any other items that relate to the requirements. Methodologies are usually vague in nature, unless applied to specific domains and use cases. The standard methodology and theory is, the module requirements should satisfy the functional requirements, and the system requirements be satisfied by the functional requirements, etc. The “Module tests” performed at the integration phase is used for the verification of the module requirements, while the “Functional tests” is used to verify the “Functional requirements”, etc. as shown in Figure 3.8.

The requirement “satisfaction” methodology is a constant feature that has to be put in practice in the specific domain and at a specific abstraction level. In the

<sup>28</sup>This should not be confused with the premise of the research work described in Section 3.1. With the exception of the description in Section 3.1, the purpose is described throughout this work is to establish traceability of model artifacts to the requirement artifacts that are being fulfilled

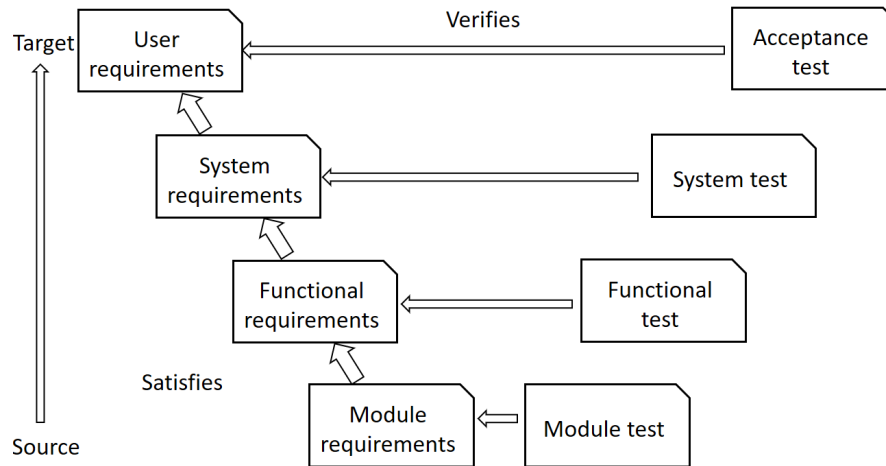


Figure 3.8: Associating requirements to other related documents

context of this work, the module design and development needs to be adapted to this methodology, within the definition of effective communication between the architect and the module developer in ASIC production. Therefore, the actual linking of the module parts to the requirements that they fulfil is desired to be traceable.

To fulfil this need, visual means of identifying the model artifacts in their relations to the requirements, like relationship graphs, hyperlinks, matrices, tables and trees are considered.

Some functional and non-functional parts of the specification can be modelled. Using the concepts and standards, and adapting some of the methodologies already developed as described above, this research seeks to combine two best of two worlds: providing a methodology for the development of ASIC by 1) using models for parts of the specification, 2) using NLs for other requirements that cannot be specified as models, and 3) combining both sets of specifications into a whole as documentation for the ASIC SoC.

### 3.10 Summary

At the beginning of this chapter, the focus was on models and standardised languages of modelling of systems. Soon after, NLs as a vital instrument of communication in systems development was discussed. Same NL was subsequently x-rayed for its special role in the inconsistencies propagated during development lifecycle of systems. It was established that it is especially the case in development environments that are heavily dependent on traditional document approach. Enabling technologies and methodologies were also discussed as part of an emerging trend to better systems development processes. Also depicted are some of the standards that are in practice. This chapter also explicitly answers the question of what requirements traceability actually is. Which further contextualises this work on its set aims.



## 4 Concepts

This chapter conceptualises the proposal for a methodology for ASIC development. During the course of the literature review done for this research, some methods that could provide useful contributions to traceability in ASIC development were discovered. These methods were collated and are explored in order to adapt them to the domain of ASIC systems development, and they are reviewed hereunder.

---

The challenge of tracing model artifacts to the requirements they fulfil is not new, as deduced from the preceding chapters. Various challenges experienced in different domains, and the attempts at overcoming the challenges, have resulted in the evolution of methodologies, standards and general advancements in the area systems development. Although these challenges are not peculiar, some of the methodologies, systems and tools are domain specific. A typical example is the software-oriented nature of UML. Furthermore, it has birthed the development of SysML, which is more for system-of-systems, including Hw-Sw systems.

SysML is still evolving, though, and some of the parts necessary for some systems' abstraction levels are already available. Notwithstanding, not all systems are being modelled with SysML. Nevertheless, the SysML profile is customisable. It allows the development of a set of customised artifacts that can be used as representation of some domain-specific components that serve as parts of a system.

Some systems and tools will be used for this work. They will be integrated into a tool-chain. The methodologies that are expected would be dependent on the individual systems used to realise the tool-chain. Therefore, this work explores the standards and technologies that these systems and tools are built on. It is assumed that the individual systems and tools to be used are compliant with the standards and technologies, which have already been discussed in Chapters 2 and 3.

To develop methodologies for ASIC development, the knowledge gained from reviewing those standards and technologies, and their applications to the area of systems design and development is to be relied upon. Of particular interest are the methods already profiled for embedded systems and model-based developments in Chapter 3. The know-how that will be transferred and applied to this work will combine some of the reviewed methodologies and extend them to adapt a working solution to the research objectives, based on the standards and technologies.

One of those standards is that of OSLC discussed in Section 3.8. It has been described as ensuring that devices, services and systems are not bound to the vendors.

Since OSLC declares that different products can be integrated into a single tool-chain, depending on what types of services they consume, this work will test the faithfulness of this declaration. Particularly, the assurance that tools belonging to the same vendors can read and consume deliverables amongst themselves. In this work, the provider, resource, resource shape and data integration defined for use in the integration of systems will be tested.

The solution also considers the SysML standards of artifacts representation, on which the models are based. It seeks out ways of identifying what the relationships between the behaviour and structure of SysML components are. This will enable definition of a mapping strategy that will be true for all mapping cases - whether behavioural, structural or operational.

Section 1 introduced the automotive sector as comprising of multiple tiers. The interaction and inter-dependence of the products of these different tiers was also briefly discussed. Also mentioned is the need for the protection of intellectual properties. Furthermore, the concept of SCM and VC was discussed in Section 3.6.3. Protecting of intellectual properties and the need for VC will be explored in this work, as a method for security feature implementation. This will be based on access control mechanisms that can be deployed as a means of controlling access to features of the collaboration platform.

Tree structures have always been used to show associations between different entities, whether in computer or contemporary sciences, or in engineering. The use of such structures is usually accompanied by arrows to show the direction of the relationship. They also show dependencies and hierarchies between connected elements of the trees. This is often helpful to show roots, stems and branches of the entities. In its approach at using basic understanding principles of life use as inputs into RE process, the proposed methodology will apply this tree and association method. While considering tree structures, attention is paid to the phenomenon of “many-to-many” [77:183], which describes the potential of growth of requirements. Many-to-many tracing could become cumbersome and unreadable.

Similarly, the use of lists, (vector) matrices and tables as traceability methods have been known to provide a helpful guide towards the understanding of associations between components and systems. These methods are solutions also applied in the everyday life of an individual, as well as in sciences and engineering. The focus of this work, in this regard, is on providing a helpful structure of these methods. This is geared towards providing a best-case scenerio for their use in ASIC development.

This research asks questions about the relationships between requirements and models. However, the search for a solution leads to the discovery of more beneficial systems and methods. The discovery provides a better relationships between design and implementation, emebded in the collaborative ALM technology. The full benefit of ALM and the efficiency it brings to developement across all platforms

is explored and the advantage is highlighted. The use of ALM in this work leads to the discovery that, names assigned to domains and the contents of the domains are important in bringing the necessary efficiency and organisation to the documentation and traceability in a production system. Traceability aside, the verification and validation activities described for DBSE (Section 3.3) is reviewed in the context of ALM. The efficiency that characterises the benefit of the use of ALM will be evaluated. This comparison will be highlighted in the subsequent chapters of this work.

The use of hyperlinks is also another valuable means of traceability. Furthermore, on the one hand is, what the tools can already do. This speaks to how compliant the systems and tools are with the relevant norms and standards they depend on. And, on the other hand, what the systems and tools should do. Here, the question is about, what is expected of the systems and tools of implementation. The test is for how well they can do what they are expected to do. This expectation should be in compliance with the relevant norms and standards they are built on. The focus is on the ALM concepts implementations (Sections 3.5, 3.6 & 3.7) and the OSLC standards (Section 3.8).

Lastly, this conceptual stage is a preview of the concepts stated here, as traceability mechanisms in RE process for a model-based ASIC development. The subsequent chapters will bring these concepts into a collective-whole as a proposal for an applicable and verifiable means of stakeholder collaboration. This collaboration will use the processes, systems and workflows that are already in place. The implemented solution will then extend the traceability options based on the applicable standards and technologies.

# 5 Implementation system

In this chapter, the discussion is narrowed down to tool-specific implementations of the standards and technologies that were discussed in Chapter 3. The properties of these tools that are relevant and applied to this research are also profiled. The specific methods which were already introduced in Chapter 4 are x-rayed as specific implementations of these systems and tools.

---

The idea behind ALM (PLM and SDLC also) was introduced in Chapter 3. It was also mentioned that different scholars have referred to it as things like a process, system, etc. The International Business Machines<sup>®</sup> (IBM<sup>®</sup>), on their part, has built their own adaptation. This is called the IBM Rational solution for Collaborative Lifecycle Management (CLM)<sup>29</sup> [78]. CLM is a system that provides variations of the systems discussed in Chapter 3 for use in managing the processes described in Chapter 2.

The concepts of Chapter 4 were implemented using IBM<sup>®</sup> Rational<sup>®</sup> set of systems and tools. The following is a discussion of these tools and their implementation as a tool-chain.

## 5.1 IBM<sup>®</sup> Rational<sup>®</sup>

International Business Machines (IBM<sup>®</sup>) Rational<sup>®</sup> (IBM Rational) is described as providing

*“a rich set of capabilities delivered as part of a collaborative and integrated range of products. Rational software is powered by Jazz<sup>30</sup> technology and supported by proven best practice processes and an entire ecosystem of IBM and partner capabilities”* [79].

The Rational platform provides a framework for integrating applications for systems and software development. It supports DevOps activities across multiple platforms and Operating Systems (OS). The applications are accessible via desktop apps or through web interfaces, depending on the type of application. Where collaboration between diverse teams is required, it is important to provide a set of tools base that can help the teams capture the different roles and tasks necessary for a successful

---

<sup>29</sup>The CLM version 6.0.6.1 is used in this work.

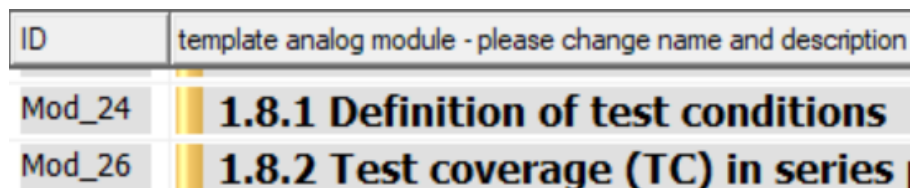
<sup>30</sup>Jazz refers to Jazz<sup>™</sup> Team Server (JTS; see Section 5.1.5)

collaboration. The IBM Rational platform provides applications and tools that enables a wide range of collaboration for diverse teams. Some of the applications within the Rational<sup>®</sup> platform, which are of interest in this work, are described below.

### 5.1.1 DOORS<sup>®</sup> Next Generation

Dynamic Object Oriented Requirements System (DOORS<sup>®</sup>) - simply referred to as DOORS<sup>31</sup> - is a proprietary tool of IBM. It is a database repository for documenting and managing requirements during a product's entire lifecycle. DOORS is primarily a desktop application management tool.

DOORS<sup>®</sup> Next Generation (DNG) is an improvement on DOORS. DNG is a web browser-based system. Similar to DOORS, DNG is used for RM and documentation. It also provides interaction mechanisms with artifacts located in other Rational tools. As described by IBM, traceability links are provided to plans, designs and models that help organise requirements into collections, modules and views for re-usability [80]. DOORS and DNG provide what is known as “context views”, as shown in Figure 5.1.



ID	template analog module - please change name and description
Mod_24	<b>1.8.1 Definition of test conditions</b>
Mod_26	<b>1.8.2 Test coverage (TC) in series</b>

(a) DOORS



445672	REQ02 - Battery Protection	<b>2.1 The Charge Controller</b>
445673	REQ03 - Adjust Charging Rates	<b>2.2 The Charge Controller capacity.</b>

(b) DNG

Figure 5.1: Comparing the context views of DOORS and DNG

Figure 5.1 are snapshots of DOORS and DNG context views. Figures 5.1(a) and 5.1(b) show item labels “1.8.1 and 1.8.2” for DOORS and labels “2.1 and 2.2” for DNG. These are subsets of items “1.8” and “2”<sup>32</sup>, respectively. Figure 5.1(a) also show identification numbers (IDs) for DOORS as “Mod\_24” and “Mod\_26”, while Figure 5.1(b) shows “445672” and “445673” as IDs for DNG.

<sup>31</sup>Unless otherwise stated, the version of DOORS referred to in this work is 9.6.1

<sup>32</sup>“1.8” and “2” are not visible in the Figure. Also, Figures 5.1(a) & 5.1(b) are the context views of DOORS & DNG, respectively

DNG provide the following six broad groups of links to consider:

**Group 1: Synonym:** in this category, the requirements are traceable to other requirement(s), but as the name suggests, it is a synonym of the requirement it is linked to. It is possible to link in any direction.

**Group 2: Embedded:** here, the requirements are found within the context of other requirements, hence the term “embedded”. They are not explicitly stated, but are a part form of the requirements to which they are linked.

**Group 3: Extracted:** the requirements that are traceable in this manner are those derived from the requirements they are linked to (in any direction).

**Group 4: Linked:** for this category, the requirements are traceable to other requirements that may or may not have been defined for them. They are, generally, representative of the previous three groups mentioned above them.

**Group 5: Parent/child:** the requirements that fall within this category are those that have been broken down into parts (children) to make the whole requirements, or have been grouped into a whole (parent).

**Group 6: Architecture element:** these are the requirements that can be traced to model artifacts that fulfil them. Within the context of this work, these are the requirements that are of primary interest. They are linked or traced to an architecture element in the model that directly implements them.

### 5.1.2 Rhapsody<sup>®</sup> (RR)

IBM<sup>®</sup> Rational<sup>®</sup> Rhapsody<sup>®</sup><sup>33</sup> (RR) is a desktop-based tool for modelling of hardware and software systems’ behaviours and parts. It is a collaborative design and development tool that is used in the creation of model artifacts that are based on specific modelling language standards. It supports UML and SysML (also a combination of both), AUTOSAR, Motor Industry Software Reliability Association guidelines for C and C++ languages (MISRA-C & MISRA-C++), etc. It allows the creation of profiles for domain-specific languages. It is also used for the creation, simulation and testing of “real-time or embedded systems and software” [81].

Relating to the context of this work, RR is used for modelling of ASIC requirements, and the model artifacts that fulfil those requirements. A model artifact is traceable to the specific requirement that it fulfils. Using RR as a desktop-based solution for requirements modelling precludes any collaboration, as only the instance managing the requirement (i.e. only the computer on which it is installed, and the model in which it is included) can access it. This is antithetical to the whole idea of MBSE.

---

<sup>33</sup>Unless otherwise stated, the version of Rhapsody referred to in this work is 8.4

### 5.1.3 Team Concert<sup>®</sup> (RTC)

IBM<sup>®</sup> Rational<sup>®</sup> Team Concert<sup>®</sup> (RTC) is an implementation of SCM practice. It uses the IBM's CCM system for tracking changes and for the delivery of work items to different tools in the IBM Rational platform. Works items are artifacts used to describe and manage a set of activities in a development project. Work items available in IBM's CCM include, "Defect", "Epic", "Story", "Task", etc. It is used for the exchange of information directly within the context of a work in progress. It provides a mechanism for automatic notification of enhancements and requested changes by teams and groups to members of the teams and groups. The notification are designed as a chat-like sessions, and include links to the affected artifacts [82]. This helps in the tracking of information changes related to requirements, especially when many people have to contribute towards a development project. During collaboration, it is important that the team members are informed of the most recent project-related information available, even if it is not the most correct. It is also necessary to be able to track the progress of items like tasks and defects. RTC provides such mechanisms, in conjunction with other tools available within the Rational<sup>®</sup> platform.

The RTC also has an extension called the "RTC Client". It is a desktop-based integration platform provided for enabling the use of RTC-based resources and functions for the local workstation on which it is installed. Although it is referred to as a "client of RTC", it is capable of standing alone as an extraction of the RTC, and can provide RTC-based services.

### 5.1.4 Rhapsody<sup>®</sup> Model Manager (RMM)

The IBM Rational Rhapsody Model Manager (RMM) offers the linking of requirements artifacts to model artifacts. It provides a mechanism to link model artifacts further with work items in CCM and QM, using the Jazz<sup>™</sup> Team Server and RTC platforms. It is also a web-based application that offers collaboration, because of its association with RTC. Thus, the models and other lifecycle product artifacts are available to stakeholders through web-interface and SCM applications [83]. RMM provides SCM feature, much like the CCM tool. The SCM feature in RMM is actually built on the SCM feature of RTC [84].

RMM is an application administration system that enables models and model artifacts in RR to be visible over a web-based client. It provides "*OSLC linking between architecture elements*" [85] and other elements in other Rational tools. Architecture elements are items designed in RR. RMM is representative of the Architecture Management (AM) in the Rational platform.

Two of the most important objects available in RMM are:

**Component:** A component in RMM represents the repository that collects model artifacts into groups. The model in a RR project that is saved to RMM

will appear under a component. The same model can be saved to multiple components, and will represent different versions of the model.

**Stream:** In RMM, a stream represents a repository in which a collection of components are grouped. Streams can contain one or multiple components, but a stream cannot contain two components of the same name.

Figure 5.2 is used to illustrate the relationship between streams and objects in RMM, and to depict how their relationship contributes to collaboration.

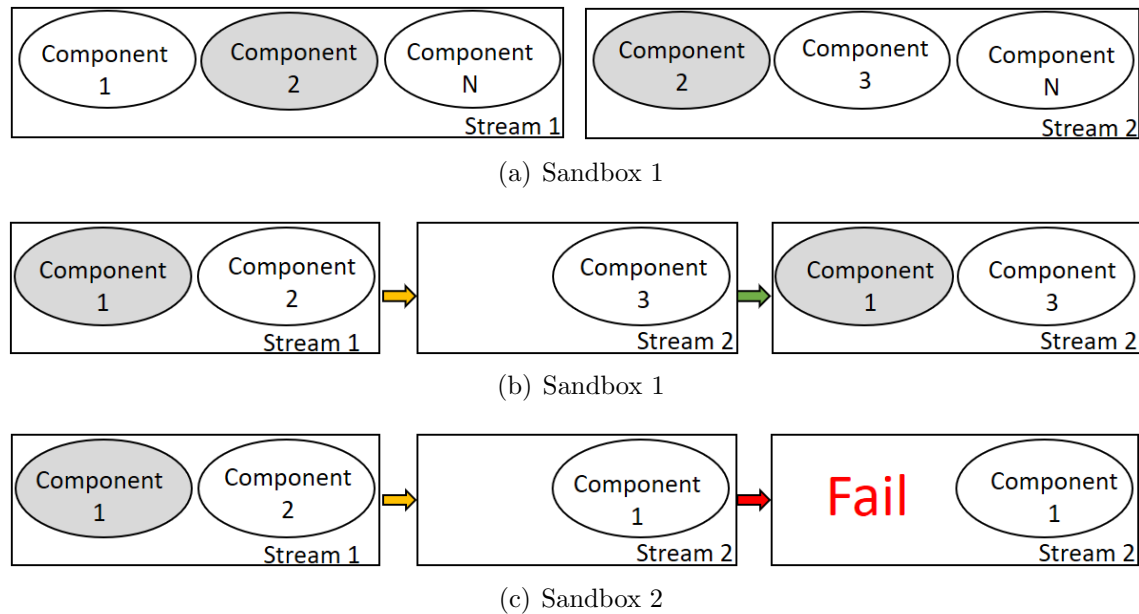


Figure 5.2: Illustration of components and streams

Figure 5.2 is a graphical illustration of the concept of components and streams. Figure 5.2(a) shows “Component 2” as belonging to both of Streams 1 and 2. In Figure 5.2(b), “Component 1” was added to Stream 2, and it was successful, but in Figure 5.2(c), the addition of Component 1 to Stream 2 failed, because Stream 2 already has a component named “Component 1”.

### 5.1.5 Jazz™ Team Server

“Jazz™ Team Server (JTS) is a Java™-based web application that runs on an application server [...]”<sup>34</sup> [86]. It provides the framework on which the RM (DNG),

<sup>34</sup>Recall that ALM was discussed as a state of the art “technology” in Chapter 3. However, any mention of ALM in this, and in subsequent chapters, unless otherwise stated, is a reference to the Bosch implementation of the ALM technology. That means, from here onwards, any mention of ALM server, if not footnoted and stated to the contrary, is a nomenclature of the Bosch implementation of the ALM technology on which JTS runs.



CCM and QM tools of the Rational platform are implemented. All of the applications registered and configured under the same JTS instance can communicate and interact with one another. It serves as a repository within which multiple project areas can be maintained, so long as they are defined in the same Jazz area. The vendor describes an “area” as a “[...] repository where information about one or more software projects is stored” [87]. A project area is defined by the deliverables expected from the area within the context of the “team structure, process and schedule”.

*“Jazz Team Server includes an extensible repository that provides a central location for application-specific information. A single repository can contain multiple project areas and their artifacts”* [86]. To enable JTS, a relational database from amongst the following has to be used:

1. Apache Derby (open source),
2. IBM Db2<sup>®</sup>,
3. Oracle<sup>®</sup>,
4. Microsoft<sup>®</sup> SQL Server.

## 5.2 Traceability in RR

Requirement diagram was developed to give systems’ engineers a view of the relationships between a requirement and other artifacts, including the relationship that exists between requirements. For example, Figure 5.3 is the requirement diagram for the “Report System Status” from Figure 2.6.

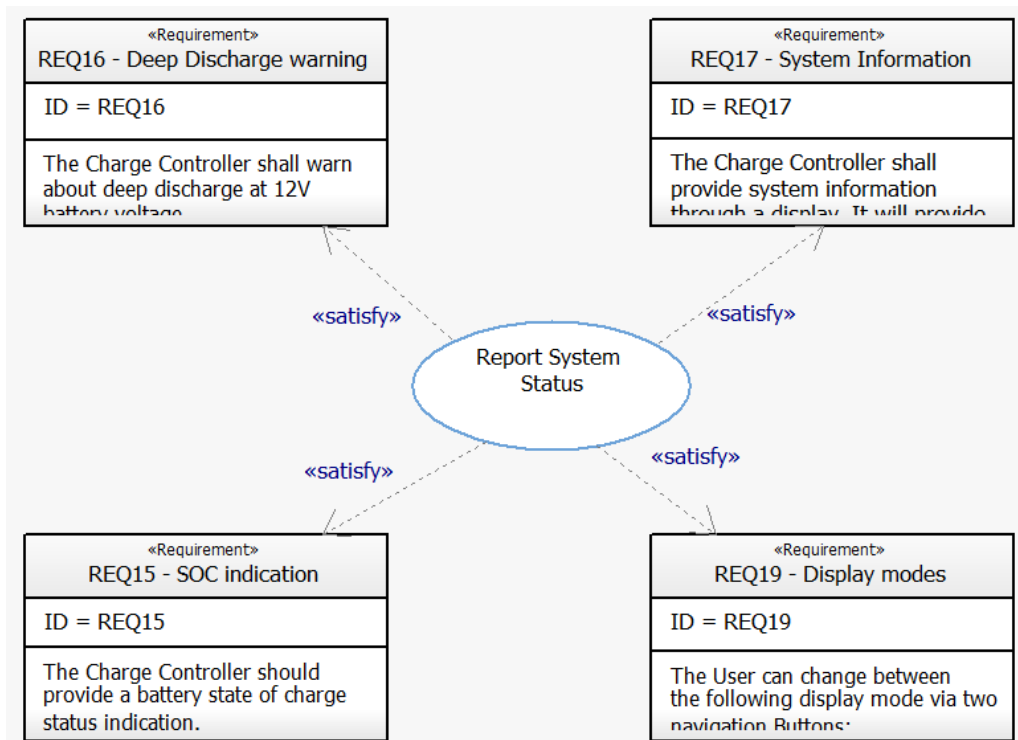


Figure 5.3: Sample view of a RR Requirement Diagram based

It shows the “satisfy” relationship between the UC “Report System Status” and the four requirements. These requirements, with IDs “REQ15”, “REQ16”, “REQ17” and “REQ19” respectively are to be fulfilled by the UC seen in the figure. As already mentioned in Section 3.9, it means the UC must satisfy each of the four requirements. This helps the teams that have to trace, verify and validate the systems to focus the tracing, verification and validation activities on specific system targets. On a small scale, this is very handy to use for traceability. But, when the requirements are many, it is inefficient to use the requirement diagram for traceability.

Section 4 mentioned the concept of many-to-many. It alludes that, many-to-many could potentially be unreadable. When the diagram is unreadable, it is in effect, visually untraceable. This is captured by the requirement diagram in Figure 5.4.

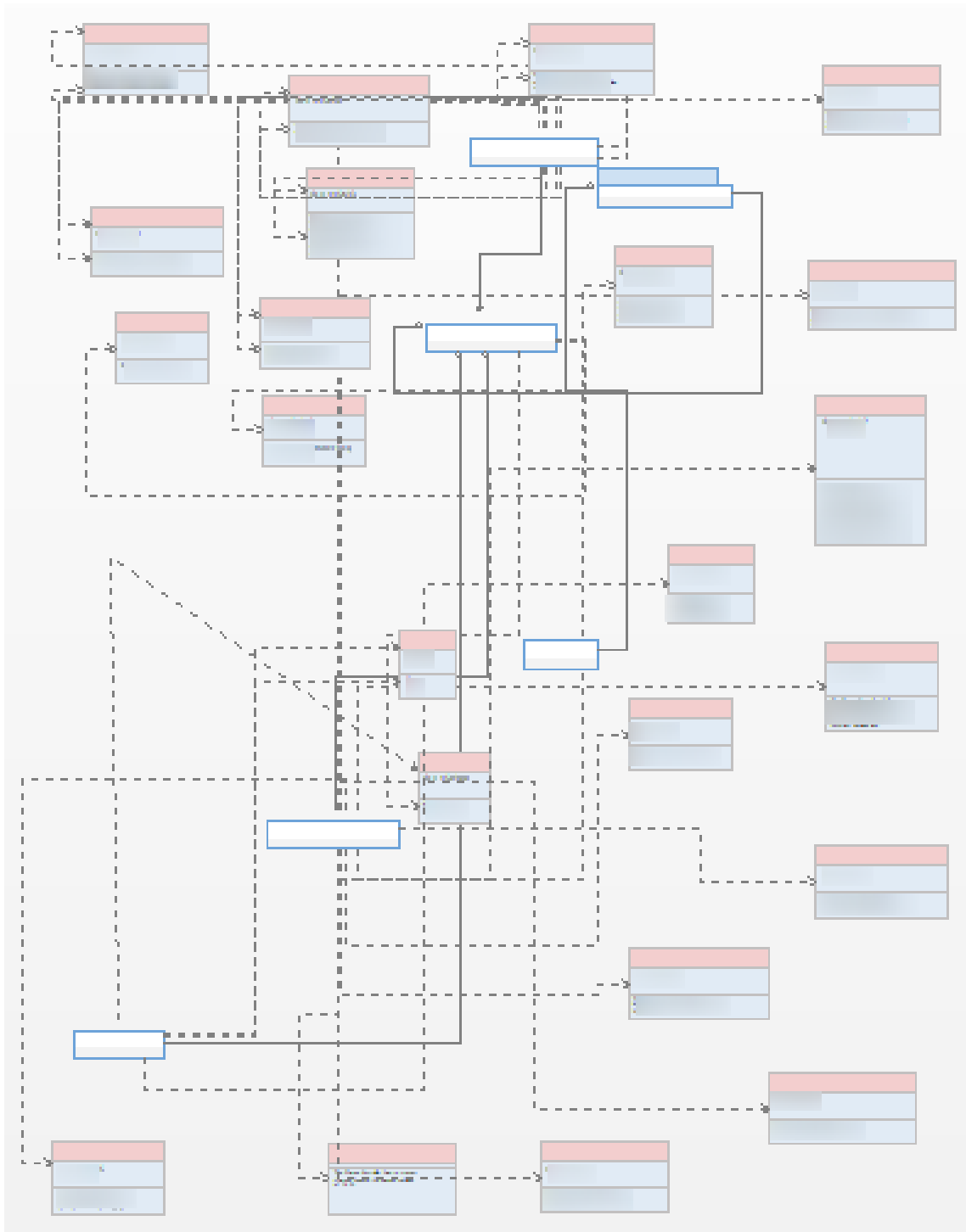


Figure 5.4: Sample of unreadable requirement diagram

Figure 5.4 shows a requirement diagram with twenty-three requirements present. The relationships is neither clearly visible nor readable. The diagram had to be minimised to 25% of the normal screen size to enable the display window contain

the depicted image in Figure 5.4. Hence, where requirements of single system's components are as many as twenty, or more, as is the case in ASIC development, the use of requirement diagram is not a sufficient traceability mechanism.

Requirement diagrams are supposed to help make the traceability of requirements easier, more efficient and pragmatic. But the preceding section shows how that aim is not met where multiple requirements are used. A handbook for methodologies based on Rational tools is the best guide. While a previous version of the book - the Release 3.1.2 - considers the use of requirement diagrams as the traceability mechanism, the current version - Release 4.1 - uses "Tables" instead [88].

### 5.2.1 Requirements tables

Tables are a more pragmatic mechanism available in RR for the tracing of requirements. With tables, a configuration can be chosen to set the table view as desired. Several options are available for this setting.

Tables in RR provide "Context Patterns". These are "a comma-separated list of token" expressions, used in describing the paths of elements in a model [89]. Context patterns become available when the "Properties" of the table layout is modified as shown in Figure 5.5.

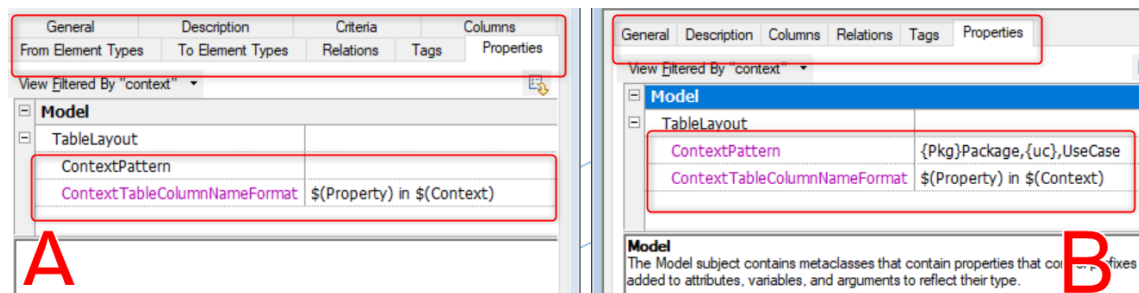


Figure 5.5: Comparison of context patterns in table layout

The right side of the figure (labelled B) contains an image of a table layout whose properties tab has been filtered for context and modified by adding the context pattern:

$$\{\text{Pkg}\}\text{Package}, \{\text{UC}\}\text{UseCase}$$

The above context pattern identifies packages (as "Pkg") and use cases (as "UC"). This sets Pkg and UC as tokens 1 and 2 respectively, to use as options in configuring the structure of a table for Pkgs and UCs, respectively. In comparison, the left side (labelled A) has not been modified. Context patterns are used in the definition of table layout properties as:

$$\text{Model:TableLayout:ContextPattern}$$

To set a defined table view, a table layout is required. The “Table” determines the actual table content and the scope of the available data in the RR browser, but the “Table Layout” is used to define the view-structure the table would have, including what type of elements and columns the table would have. Tables show the rows as model elements, and the columns as information about them. They can also be used such that, the first columns display available model artifacts, grouped or ungrouped, and the succeeding columns provide other information about the artifacts in the first column. Filters can also be set to display selected table contents in the cells.

### 5.2.2 Matrix views

Another valuable mechanism currently available for the processing and analysis of requirements’ relationships with other artifacts is the “Matrix Views”. Like Tables, matrices require the use of a “Matrix Layout” to define the structure and element types the matrix view would be populated by. Furthermore, the views can be scoped and set to consider the hierarchies of the displayed items. Matrix views can be switched to alternate the columns and rows. That is, rows become columns, and vice versa, which is a property not available to tables in RR.

Matrix views, however, can be very cumbersome and information contained in the cells, although comprehensive, are not easily trackable. The reason they are difficult to track is, like requirement diagrams, they can grow potentially out of scope of the computer screen. Furthermore, unlike requirement diagrams, they cannot be filtered from the views of the matrices. Worse still, the first column cannot be adjusted in real-time to reduce their sizes, so it covers much of the display window available for viewing the RR model artifacts contained in the matrix.

### 5.2.3 Annotations, relations and tags in RR

There are several model artifacts available in RR that help contextualise the requirements and model artifacts. Some of which have been reviewed thus far. Those are the relationships between RR items and other rational tools. This group of model artifacts includes, dependencies, derivations and satisfaction, all of which are related to “architecture element” (described in Section 5.1.1).

A hyperlink to other model artifacts can be set from a model artifact in RR. For example, a requirement diagram could be hyper-linked to a use case diagram, to provide the relationship context within the model. Although hyperlinks are very useful, they do not provide a compact, concise and comprehensive view of the contexts and uses for the designated model artifacts.

Annotations include “constraints, comments and controlled file”. Comments are traditional means of adding contexts to the requirements. They are not actionable, that is, they cannot be executed. However, they help clarify the context or usage

of a model artifact, or the purpose an addition or modification of an item serves. Using comments is handy for one, or a few cases. It becomes impracticable to be deployed in models for a thousand-plus ASIC requirements.

Constraints and controlled file were not reviewed for this work.

Tags are used to provide additional, and importantly, actionable information on model artifacts. With tags, performance metrics, like minimum and maximum values can be supplied to system parts. These are then applied to the appropriate areas of systems behaviour metrics, like in code generation or simulation.

### 5.3 Summary

The systems and tools deployed as ALM-based implementation was introduced as IBM's CLM. They were also introduced as compliant with the OSLC standards. SysML had been described in Section 2.5 as a modelling language standard. In this chapter, RR was introduced as the Sysml-based tool on which this work was dependent. The traceability methods described in Section 3.9 and in Chapter4 were also pictured as they are implemented in RR. These methods were characterised as insufficient solutions in the context of this work. The insufficiency of these methods was pictured as partly a direct consequence of the size and complexity of requirements in ASIC the domain.

## 6 Integration and deployment

In Chapter 5 above, the tools that are used in this work were introduced. This chapter will apply these tools to as much as they support the standards with which they are compliant. ALM has been introduced as process, standard and technology-based. Although these tools are vendor-specific, if they are compliant with their standards, the application will produce similar or comparable results as any other tools based on these standards. Thus, a tool-chain comprising of these standards will be realised at the end of this chapter.

ALM is essentially, a concept of data integration. Different implementation of this concept will vary, according to the constructs perceived by the designer. For this reason, although systems and tools may follow standards, the implementation of the standards may differ amongst vendors.

The first step towards integrating the tools described in Chapter 5 was to understand the nature and structure of their design. Thus, they were firstly separated into different Project Areas (PAs) and zones, in line with their design [87]. This is illustrated using Figure 6.1<sup>35</sup> below.

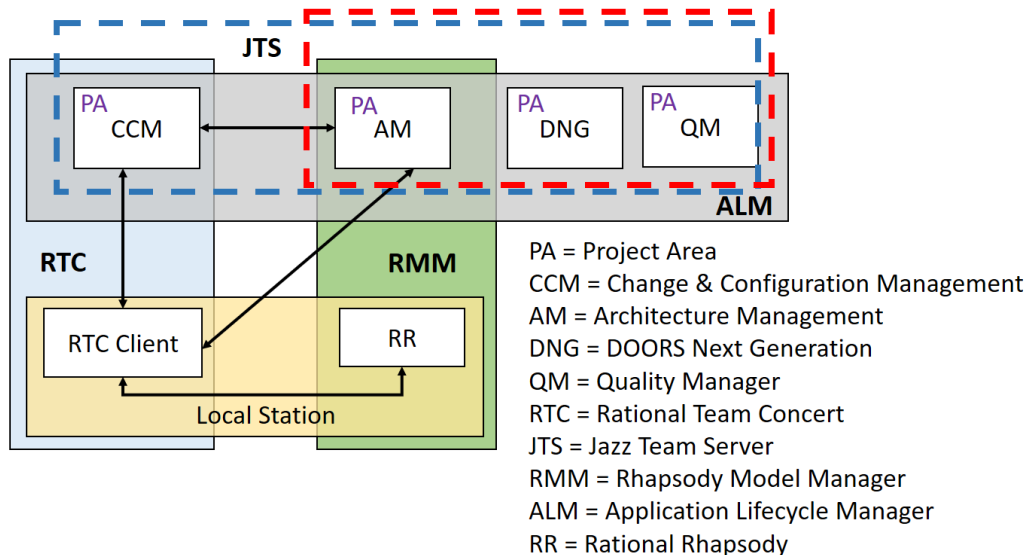


Figure 6.1: Structure of ALM

<sup>35</sup>ALM referenced here is the setup at Bosch. The name may differ for other organisations, but the structure is consistent.

As depicted in Figure 6.1, each instance of the AM, CCM, DNG and QM represents a PA<sup>36</sup>. Each is also part of the same JTS instance. The yellow-coloured area depicts the workstation where RR and the RTC Clients are installed. They also belong to the green- and blue-coloured areas, which represent the RMM and the RTC respectively.

However, this did not provide a complete picture for integration, as it does not provide a clear understanding of the component-parts of each system's role in the integration. Therefore, the system was further categorised into “physical and logical” structures of a data communication and integration platform. The following images in Figure 6.2 were the communication structures pictured of the system.

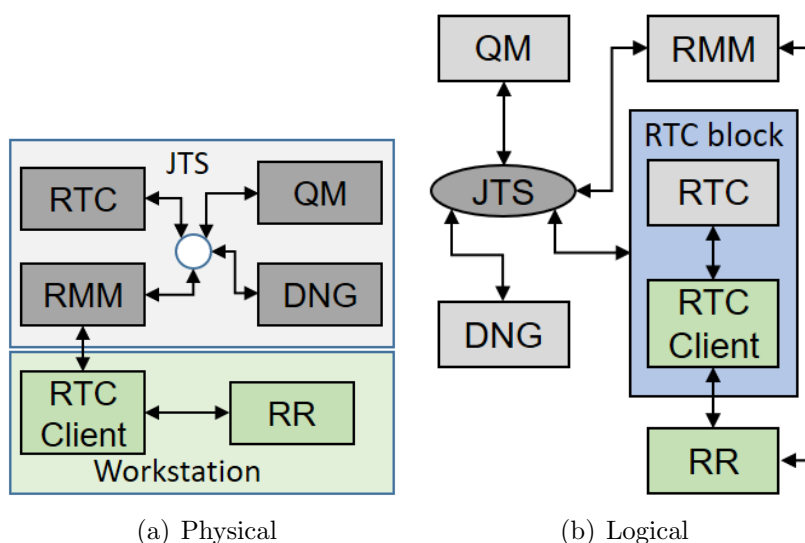


Figure 6.2: Structure of JTS-based platform for integration

In Figure 6.2, the entire communication framework is pictured as divided into physical local and online server-based infrastructures. RTC, RMM, QM and DNG all belong to the same JTS server instance. The RR and RTC Client are local installations on a workstation. However, Figure 6.2(b) shows the same tools as entirely based on logical role each plays. It also shows “RTC block” which includes all the services (e.g. CCM) provided by the RTC. The physical structure details the interconnectivity between the infrastructures, although the communication that exists amongst them are functions of the logical framework. From Figure 6.2(b), it can be seen that JTS can communicate with each component-parts of the RTC block, independent of the other. This is interpreted as, “1) authentication and access to RMM, RTC block, QM and DNG is provided by JTS; and 2) the communication of DNG-based information between RMM and RR is possible, so long as RR is enabled and active RTC Client”.

<sup>36</sup>AM and CCM are synonymous with RMM and RTC, respectively. Thus, they are used interchangeably, depending on the contexts. However, each refers to its same respective concept.



With this picture ensuring the clarity of the structure and roles of the systems, instances of PAs were created for project developments as PAs. Several options are available for the creation of these PAs. However, a suitable option to use a lifecycle PA was chosen. Although each can be created independently, each instance of DNG, RMM, RTC and QM were created together as a lifecycle PA.

As already mentioned, all the PAs within a common JTS instance can be integrated. This is shown in Figure 6.3<sup>37</sup>.

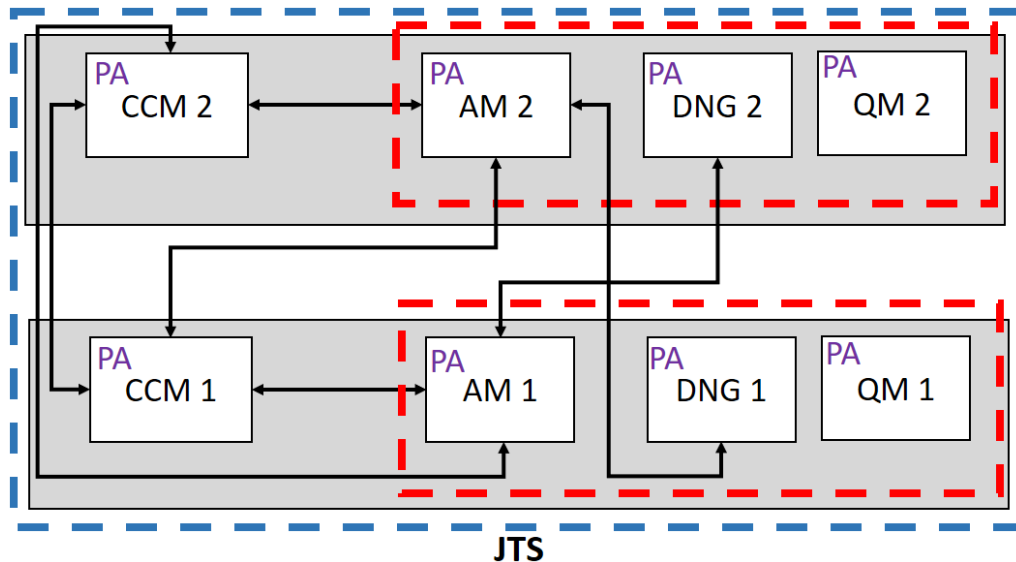


Figure 6.3: Sample structure of multiple PAs

The PAs in the schematics are parts of the same JTS instance. CCM 1 uses components available in CCM 2 and is added as a PA to AM 2, as shown in Figure 6.3. CCM 2 also uses components from CCM 1 and is reciprocally added as an AM 1 PA. DNG 2 (i.e. the requirement artifacts in it) is also re-used by AM 1 PA. The lifecycle PA is configured for deployment as a RMM-enabled system. Common accessibility to the PAs is a JTS feature, so authentication is ensured across the entire platform.

This property of the assembled system was used as a basis for the integration of the tools as a tool-chain. The integration was stepwise, starting out with the use of DNG and RR integrated into the RMM development platform through the RTC Client. Further integration included the RTC and the QM PAs.

## 6.1 Linking the project areas (PAs)

As previously described, the primary aim of this work is in relation to the handling of ASIC product requirements available in DNG. This includes establishing traceability

<sup>37</sup>Although ALM here is referenced to the setup at Bosch, it is typical of Application Lifecycle Management in general, including the CLM of the Rational platform.

between the requirement and the model artifacts. The repository of the model artifacts is the AM (RMM), although the models are to be designed and modified using RR. Considering the complexity of the system as described above, the tool-chain used in this section is as shown in Figure 6.4.

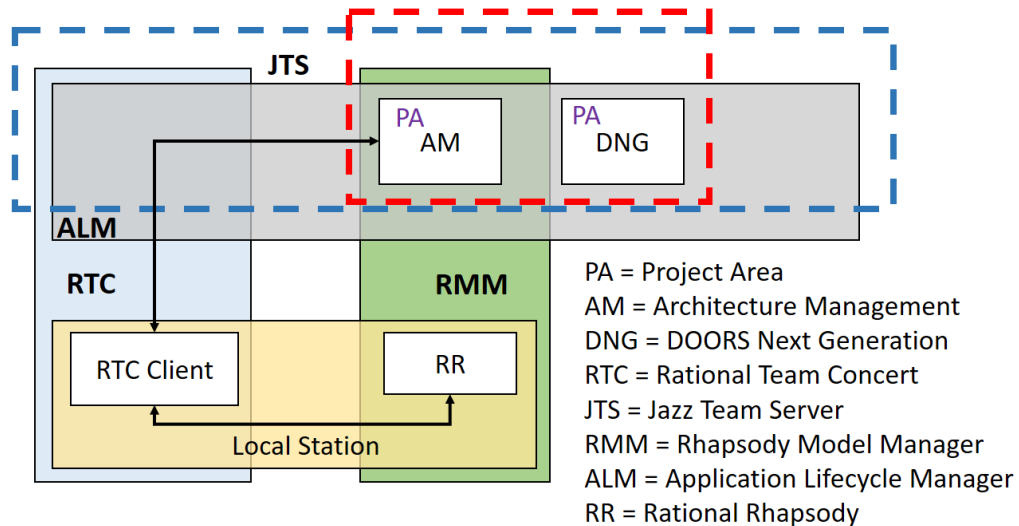


Figure 6.4: Project setup for DNG and RMM

Each instance of a lifecycle PA that was created with the inclusion of only RMM and DNG results in the structure depicted in Figure 6.4. With this structure in focus, a PA was created in the ALM server. The DNG instance that was created in the same lifecycle PA as RMM are already linked. However, other DNG instances were created and added as part of the requirements repository that provide requirement artifacts for projects in the AM. This was used to ascertain the reuse of requirements, when they are available in different PAs.

To connect RR to the RMM resource available in the JTS through the RTC Client, the resource address is provided in the RTC Client as:

```
https://<serverDomain>:<port>/am38
```

This established a communication path between the RR and RMM. The communication path of the connected PA and the local workspace is illustrated using Figure 6.5.

<sup>38</sup>“am” represents “Architecture Management” (AM) already described in Section 5.1.4

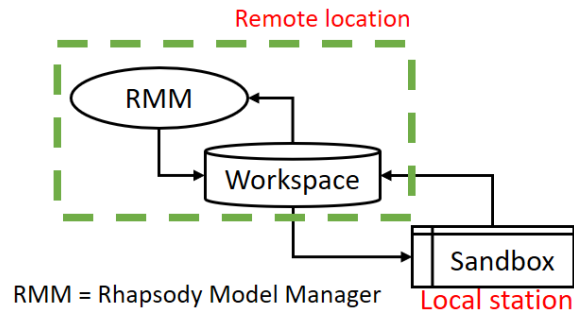


Figure 6.5: Communication path between local and remote repositories

Figure 6.5 is a schematic of the established communication path between RMM and the RTC Client. The “Workspace” and the “Sandbox” are components of the RTC. They serve as repositories for models and other work items they are associated with. Transfer of artifacts is stepwise from the sandbox to the associated workspace, then to the defined RMM. Correspondingly, download of artifacts from RMM to the workstation follows the reverse stepwise manner, as illustrated by Figures 6.5.

The default behaviour is for each sandbox to be associated with a specific workstation. Multiple workstations can also be associated with a single RTC Client by default. This is illustrated in Figure 6.6.

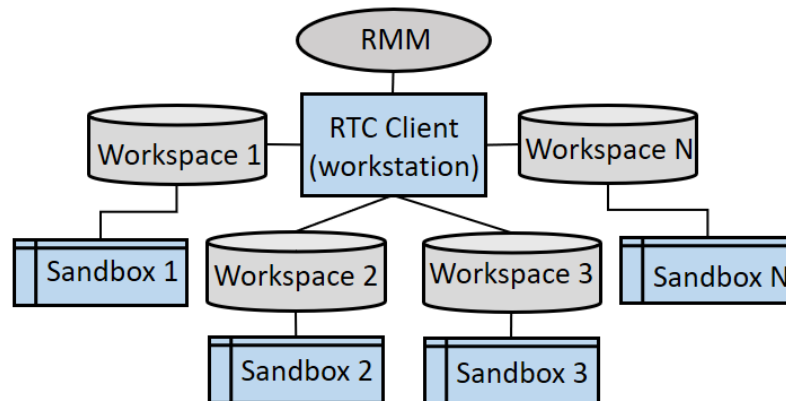


Figure 6.6: RTC Client-workspace connection on a workstation

Figure 6.6 shows a RMM linked with a RTC Client present in a workstation. The client is connected to n-number of sandboxes available to the workstation. It shows each sandbox as accessing the RMM through their common RTC Client. The RTC Client manages and maintains each connection to the RMM separately, but from the same interface. This is ensured through the use of separate locally available folders containing unique hashed data that helps associate the sandboxes to the RTC Client and their workspaces. The grey-coloured items in Figure 6.6 represent online, server-based resources, while each blue-coloured item is locally available. The system was further pictured as the physical and logical structures shown in Figure 6.7.

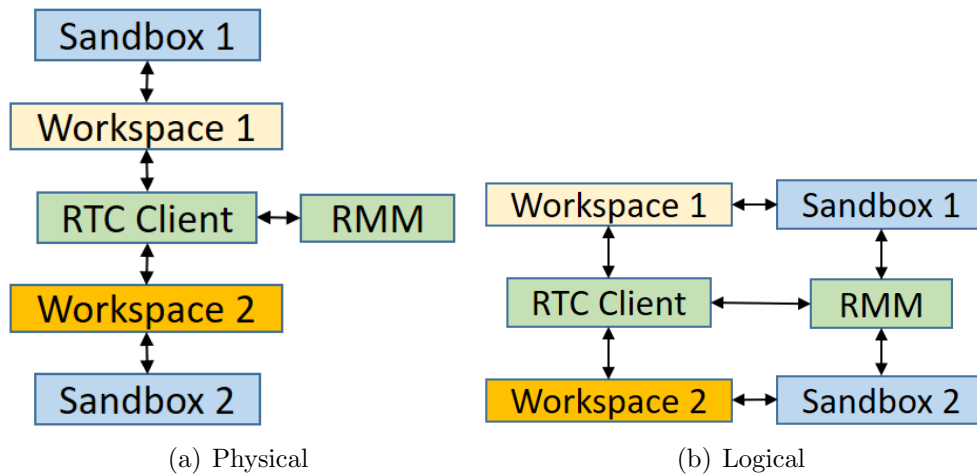


Figure 6.7: Comparing the context views of DOORS and DNG

Figure 6.7 is a system of two workspaces connected to a local installation of RTC Client. The logical connection of Figure 6.7(b) shows that the RMM has a direct access to Sandboxes 1 and 2. However, this is only possible if the physical connection in Figure 6.7(a) is available. Nonetheless, the direct access between RMM and RR is also limited to the communication of DNG-based artifacts, as already described in the preceding section.

The AM can be created as a PA belonging to a person or team, and is accessible by other selected members of the ALM server (maintained in the same JTS area). Contrarily, the workspace is a “personal data bank” of each RTC Client-connection to the AM. It is an online remote storage for model artifacts awaiting upload to the server or download to the sandbox. However, workspaces can also be made available for team use. The sandbox is also an accessory of the RTC, but is located on the workstation where the RTC Client is installed. And, like the workspace, each sandbox belongs to a specific RTC Client-connection to the AM. Unlike workspace, it cannot be made available for team or collaborative use.

## Access to the models in RMM

Models cannot be created with the RMM. The AM is only used for management and storage of model artifacts, and to enable collaboration between members of diverse SE teams who have to contribute parts that fulfil a set of system requirements. Therefore, a model was made available in the RMM for the collaborative activities. To make a RR-based model available for deployment to RMM, two methods were used:

- an existing model was copied into the sandbox, then RR was launched from inside the sandbox;
- a new model was created in RR, then saved in the sandbox.

To perform the above actions, a workspaces were created, and the sandbox associated with the workspaces loaded onto the RTC Client.

## 6.2 Adding CCM and QM to the lifecycle PA

The desire for MBSE is in part, its ability to bring designers and developers as close to the RE phase as possible, during the course of their day-to-day business of modelling [3:9]. That means, they need to be able to add and modify requirements as they deem fit. This is because they are closer to the system at an implementation abstraction level than other stakeholders involved in the RE process. And, additions and modifications made should be trackable. Also, when changes are required and to be delivered, it is desired that requests for such changes should be made as part of the development process. Such requests should be addressable to the stakeholder to enable or fulfil those requests. Furthermore, progress on such changes and any development activities spawned from such requests and changes should be available as part of the development process. This means, the requests and changes should also be trackable.

Changes and requests are embedded as artifacts in the CCM of RTC. Although, RMM has a SCM property that is built on the CCM feature of RTC as mentioned in Section 5.1.4, instances of RTC as PAs were used to ensure uniformity across the development platform. The artifacts of the RTC are known as Work Items and Work Plans. These enable tasks, reporting and tracking the status of models and general development activities.

The structure of the system at this abstraction is illustrated as the communication path visible in the schematics of Figure 6.8.

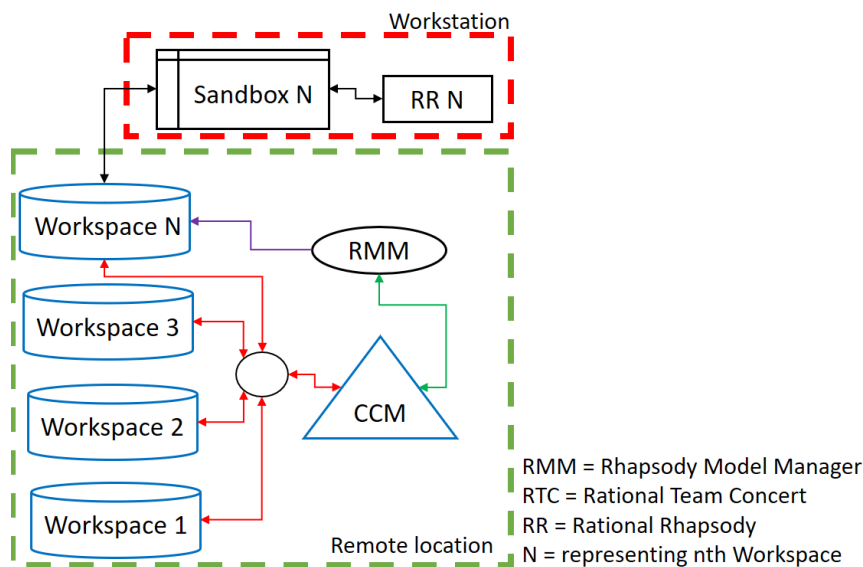


Figure 6.8: Change sets as a function of CCM

The focus here is on the role of the CCM. “N-number” of workspaces collaborate on a RMM-based project in Figure 6.8. Members of a collaboration team can create or modify existing artifacts, subject to their assigned roles. Each member is added to the specific PA from the individual management console of the named PA. Each team-member has corresponding workspaces generated for every instance of each workspace that is created and loaded a each RTC Client. Multiple workspaces and sandboxes can be created for every RMM project. However, the “change sets” created for the project are visible to all members alike, whether it be the default change-sets created by RMM, or those delivered via the CCM system.

The model in RMM is directly downloadable to the sandbox as shown in Figure 6.8, as indicated by the purple arrow linking RMM to “Workspace N” uni-directionally. This occurs when a new workspace is created and the sandbox is loaded on to the RTC Client. It can also be downloaded via the CCM. This is the case when an update is available in the RMM. The combination of green and red bi-directional arrows of Figure 6.8 are used to illustrate this case. Uploading to the RMM server however, is only possible through the CCM feature. Data exchange initiated from the local sandbox has to go through the CCM to RMM. The green arrow is only to indicate that the CCM (delivered from the RTC) can also exchange information directly with the RMM, independent of the links to workspaces.

### 6.3 The case for a naming convention

A complex set of requirements was also provided for test-running the concepts developed in the course of this work. A selection of these requirements were transferred to DNG. The requirements were titled structured and contextually. Having transferred the requirements to DNG, work items were created in CCM and were used to label, monitor and track the usage of the requirements. They were also used to track activities related to validation and verification of system’s processes. These verification and validation-related activities are provided by the QM. The QM artifacts are Test Plans and Test Cases.

The requirements entered into DNG have unique identification numbers (IDs) that are generated automatically. While some of the attributes of DNG-based requirement artifacts are modifiable, these IDs are not. The requirements themselves are portable - they can be dragged to different positions in the requirement document. The IDs and other properties also move with their associated requirements, though. The requirements are to be tracked, nevertheless. Therefore, the work items were renamed after the requirements they refer to. Where the requirement is just a “heading” to contextualise<sup>39</sup> the requirements under it, meaning that it does not contain any properties that is to be fulfilled by tasks, the CCM based work item type called “Epic” was used. This helps to put the task-sets in the same context as there are for the requirements in DNG.

---

<sup>39</sup>Contexts of requirements is as illustrated in Section 5.1.1 using Figure 5.1. It groups requirements under the same heading.

“Mock tests” were created in the QM to associate the requirements and the task sets associated with them. As already described, the task sets refer to the implementation instructions that are used to direct what activities are to be performed in order to fulfil requirements. Consequently, the mock test-plans generated in the QM system were intended to characterise the implementation of verification and validation activities, models and requirements. These tests also used naming schemes consistent with their corresponding system requirements repositied in DNG, and the work items in the RTC’s CCM.

## 6.4 Application of attributes

From Section 5.1.1 above, it is deduced that, a link in DNG is either inbound or outbound. It depicts a requirement’s relationship to any one or a combination of other artifacts. Requirements could be related to other requirements (e.g. decomposition), or to task-sets (e.g. implementation), or to architecture elements that fulfils it (e.g. satisfaction). It can also be internal, i.e. it links requirements to other artifacts within the same lifecycle PA - the same JTS, or it could be external. Relationships to other tools outside the Rational range of systems and tools is also captured using links, as described. Linking each requirement to the artifacts and systems that contribute to its realisation is, therefore, a necessary step towards ascertaining the compliance of a system with its design goals. The linking of requirements to other artifacts ensures they are been tracked. Links were also established to the tests so that the systems behaviours and functions will be tested against specification. For this reason, appropriate labelling of the requirements is required.

To label the requirements appropriately, a suitable artifact type for each requirement was defined. This is illustrated in Figure 6.9.

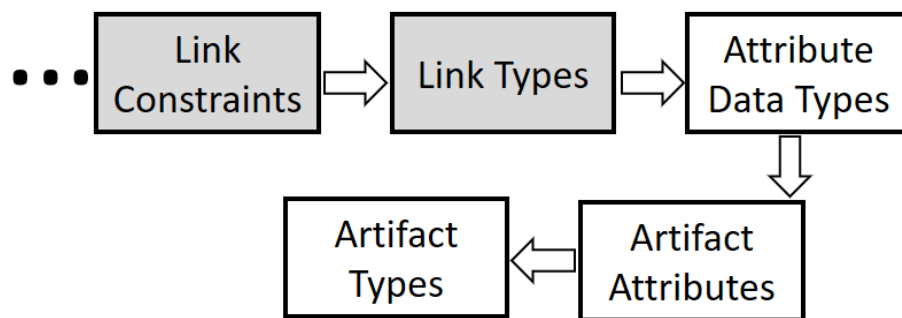


Figure 6.9: Creating the required artifact types

Figure 6.9 above shows the order of creating project-specific custom artifact types that required capturing. The grey-coloured boxes in the image indicate items that were not considered in the course of this work, and the ellipsis indicates there are other requirements artifact items preceding those displayed in the image. The data

types were used to create appropriate “Artifact Attributes”. The artifact attributes were subsequently used to create the desired “Artifact Type”.

By default, the “Data Types” listed in Table 6.1 were some of those available or created in DNG.

<b>Attribute</b>	<b>Data Type; S = System-defined, Comments</b>
ArtifactFormats	Enumerated (S), Collection, Text, Module, etc
Date	date (S), system-defined
DateTime	date time (S), system-defined
Float	float (S), system-defined
Integer	(S), system-defined
Severity Type	E, Range 0 to 5, system-defined
String	string (S), system-defined
Time	time (S), system-defined

Table 6.1: Sample of data types in DNG

Some of the attributes are “system-defined” (labelled “S”) and are, therefore, not modifiable from the management console interface of DNG. Others, such as “Test Val” were created. These data types served as inputs in the creation of the “Artifact Attributes” shown in Table 6.2 below.

<b>Artifact Attribute</b>	<b>Data Type; S = System-defined Comments</b>
Created On	Datetime (S),
Description	String (S),
Maximum Value	Float,
Minimum Value	Float,
Modified On	DateTime (S),
Requirement Type	String,
Severity	Severity Type,
Status	String,
Title	String (S),

Table 6.2: Sample of attributes in DNG

Table 6.2 is a list of some of the important artifact attributes required for processing the requirements considered. They also carry important information about the requirement artifact they are attributed to. They further serve as inputs for the creation of “Artifact Types”.

The DNG artifacts present in the two tables above provide the necessary inputs in creating the items available in Table 6.3 below.



<b>Artifact Type</b>	<b>Default Artifact Format</b>	<b>Comments</b>
Heading	Text	
Information	Text	Priority, package info, etc were applied
Requirement	Text	Maximum and minimum applied to it
Test (Plan) Collection	Collection	Attributes like priority were applied to it

Table 6.3: Sample of DNG artifact types

Artifact types are project-specific, and are created/generated according to the needs of a project. Consequently, none one of artifact types is system-defined. The artifacts can have attributes like “Requirement Type”, which is included in Table 6.2, and formats like “Float” or “String”, which are part of the attributes shown in Table 6.1. The former is an attribute, and the latter are required data types. These data types carry the information about the requirement, and determine what kind of project-dependent values they can be assigned. The data types, and attributes are then assigned to the artifacts shown in Table 6.3.

Choosing suitable icons for the artifact type also aids in conveying important information at first glance. The attribute and data types were used to plain specific constraints on the requirements developed for the project, then appropriate tasks and (mock) test plans were associated with each requirement.

## 6.5 Handling RMM-based models

Having created the models, requirements’ artifacts, work items that stipulate roles collaborators play in the realisation of a project, were deployed for integration. Some performance metrics were also defined for accessing the project, especially with respect to task schedules. Deployment of the system as a functional unit of a project, provides the framework on which the three Rational tools mostly of interest are evaluated. The three tools are: 1) RR, for making models, 2) DNG, the RM system as a repository for managing the requirements for the models and 3) RMM, the AM tool for handling the model as an online internet-based resource. Also evaluated are the other Rational tools deployed alongside the previous three mentioned. They are: 4) RTC, the CCM tool that provides configuration management of the model as a CLM project, and 5) the QM system, for testing and verifying the model’s implementation of the agreed requirements. They are evaluated starting from preparing the individual tools, as described in the sections above, to the resultant tool-chain, as is described below.

### 6.5.1 Delivery to RMM

In Section 6.1, establishing a connection between the local storage and the remote repositories for the model was discussed. And, particularly, in Sections 6.1 and 6.2, the system was described as an assemblage involving downloads and uploads. Now, in this section, the focus is on the delivery of artifacts from local to RMM.

Before any collaboration on a project is effected, a model has to be uploaded to RMM. This upload is termed as a “Check-in”. Checking in to RMM is illustrated in Figure 6.10.

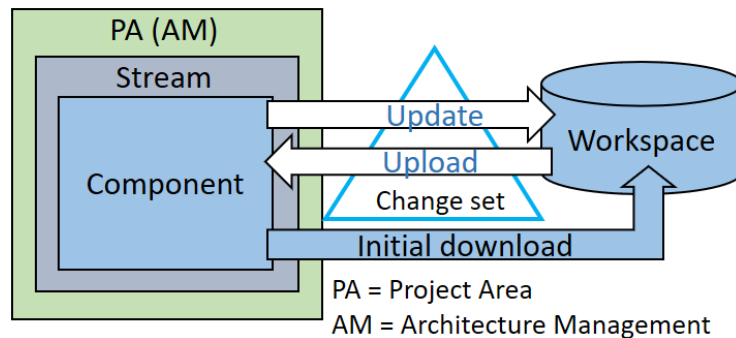
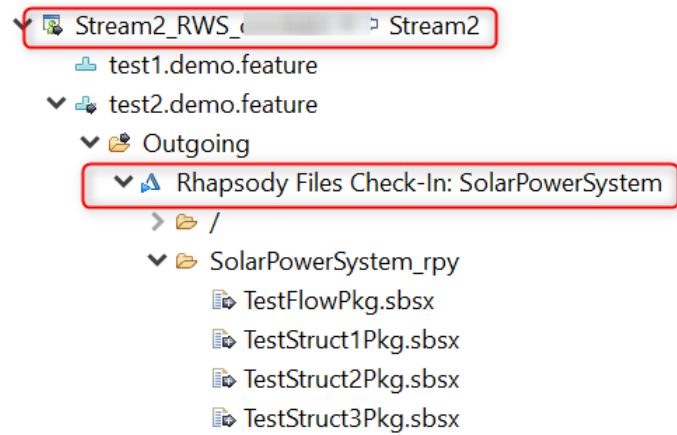


Figure 6.10: The central role of change sets

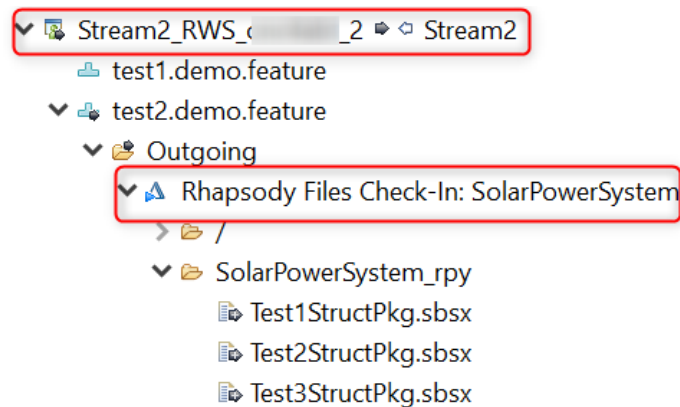
“Change set” was defined. It is needed for every checking in of model artifacts. It is also used for model downloads into the sandbox, except for the first download. This is captured in Figure 6.10 as “Initial download”. For uploads, the contents of the sandbox are copied into the “Component” available in the “Stream” of the RMM PA, as depicted in Figure 6.10.

### 6.5.2 Collaboration and harmonious modelling

With models and requirements available in RMM and DNG respectively, multiple client-connections were made to the project based on the principles and processes discussed in the preceding sections. Although multiple workspaces can exist for a single RTC Client of a given project, the default change sets, however, are attached to the named person. That is, when two workspaces are created on a client computer belonging to “John Doe”, for a project titled “Project 1”, each workspace is a separate entity named “Workspace 1 Project 1” and “Workspace 2 Project 1”, respectively. However, both workspaces use the same change set as “Rhapsody Files Check-In: Project 1”, as illustrated in Figure 6.11



(a) Workspace 1



(b) Workspace 2

Figure 6.11: Behaviour of single RMM PA and RTC Client with multiple-workspaces

Figures 6.11(a) and Figure 6.11(b) show the contents and structure of two workspaces, “Stream2\_RWS\_xxx\_Stream2” and “Stream2\_RWS\_xxx2\_Stream2” respectively. They both contain the same project in “Stream2” that has “test1.demo.feature” and “test2.demo.feature” components inside it. The name of the RR project that is available in the RMM PA is “SolarPowerSystem”. This signals that, a first copy of project SolarPowerSystem is available in workspace Stream2\_RWS\_xxx. Subsequently, a second copy of the project was created for a different workspace as “Stream2\_RWS\_xxx2”. Both are replicated copies of the same SolarPowerSystem project available in two separate sandboxes via the same RTC Client on the same workstation. Additional change sets were subsequently created, in the form of comments.

For collaboration, the model contained in each workspace was modified, and the changes sent as updates to the server. Each transfer to the server was accompanied by a message notification to other workspaces connected to the now RMM project, as shown in Figure 6.12.

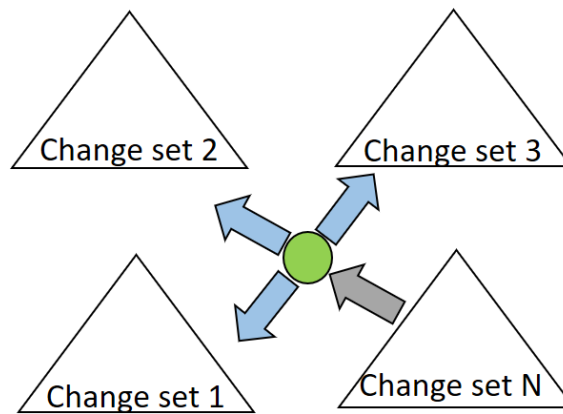


Figure 6.12: Change set notification pattern

Figure 6.12 is a visual representation of the communication format amongst the connected workspaces. “Change set N” represents any workspace from where modifications to the model is uploaded to RMM. The CCM feature is engaged by this transfer process, and the message notification is received by change sets 1 to 3. The available updates are accepted into each workspace, via their respective change sets.

Up to this point in the implementation, after the set up, only model delivery and retrieval to realise a consistent up-to-date model has been implemented as part of the collaboration. Collaborative modelling was further explored by adding links from the models to the requirements. Linking a model artifact to requirement artifacts was completed bi-directionally, i.e. the links were added inside RR, and delivered to RMM via the change sets, and links were also added from within DNG, and saved. Additionally, new requirements were added to DNG from RR, and also from inside DNG, and accessed via RR.

The other Rational tools hosting the remaining PAs were then integrated into the collaboration. The change sets were associated with the work items in CCM and the test plans in QM. Furthermore, new requirements were added to DNG from QM. From within the management console of RMM, “OSLC Linking” was permitted, in order to enable further integration of the entire system. Model artifacts were linked successfully to the the CCM, DNG and QM from RR. Each update to the RMM was correspondingly announced as update notification to other workspaces. When new workspaces were created and loaded, they were up-to-date as at the last known model configuration available in RMM.

### 6.5.3 Extending views and tags

After the steps described above had been completed, and the traceability links between the model, requirement, test and work item artifacts had been established, it was necessary to visualise the implemented solution in a comprehensive and concise manner. Matrix views and table were used to perform the set of operations required

to present a logical assessment of the model artifacts and their relationships to other work items.

For the matrix layout, the “Cell Elements Types” chosen was “OSLC Link”. The context of OSLC links was preferred, because the artifacts that were to be related to the model items on the view are all remote artifacts, linked together via their OSLC properties. The “From Element Types” and “To Element Types” option were varied. The matrix view was scoped to capture the model artifacts outwards from the innermost file of interest, which is depicted in Figure 6.13 below.

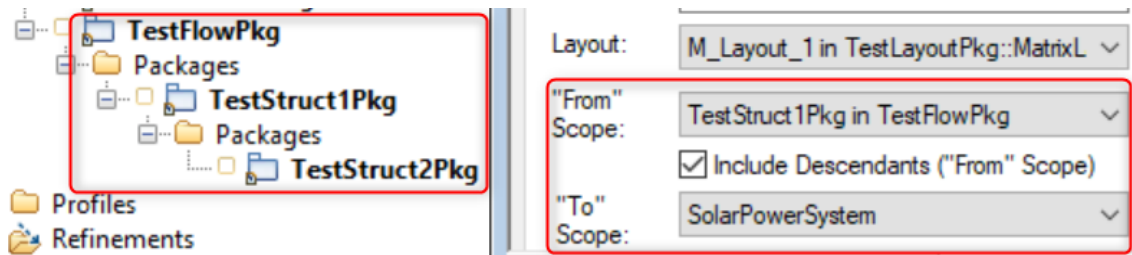


Figure 6.13: Setting the scope for table views

The example of Figure 6.13 is a sample of the features of a matrix view that shows the scoping of the matrix information for packages from “TestStruct1Pkg”, which is the innermost item in the “TestFlowPkg” package. TestStruct1Pkg was chosen to capture the requirement information for the items in the nested packages contained in the “SolarPowerSystem” project.

Tables were also developed for viewing the requirements and their linked model artifacts. A default read-only requirement table view was available in the “Remote Artifacts” section of the model. This table is populated with the contents of the associated DNG PA. The Requirements were available as a “Collection”. Five attributes of the requirement artifacts were visible on the default table. These available attributes are as shown in Figure 6.14.

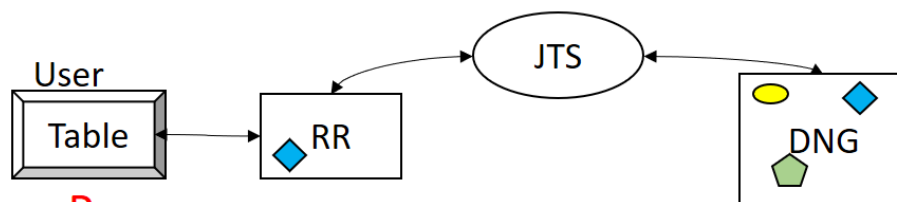
ID	Name	Specification	Link Type	Link From
450902	450902: Charge Controller Req	Charge Controller Req	Refinement	Charge_Contr...
450900	450900: System Req	System Req	Satisfaction	SolarPowerS...

Figure 6.14: Default table view

OSLC, discussed in Section 3.8, was relied upon for extending and adapting the returned information needed for creating customised tables, as conceptualised in Section 4. Firstly, the information content and context of DNG was modified by

adding two columns. These columns were named “Minimum” and “Maximum”, respectively. These were used to provide details about the requirements in DNG. Since the default requirement table populated in RR only includes the five information fields of Figure 6.14, the performance information added to the two extra columns were not available in the model. This is because it was not transferred from DNG to RRM. The local workstation and the RR were customised to request and accept the information from DNG. Firstly, the OSLC configuration was modified, and the additional information required were added to the configuration file. The file is a JSON format file containing information about the appearance and properties for other Rational tools for RR. Afterwards, RR properties were customised and instructed to accept custom OSLC information.

The informational dialogue that extends the use of tables for the traceability of model to ASIC requirement artifacts is shown below. Note that this communication dialogue are applicable at different layers. They were applied to specific projects. However, they were restricted to the specific projects on the local workstation as a block. They can also be applied to the JTS domain. Figure 6.15 is used to illustrate this communication dialogue process between RR and DNG.



### Process:

1. DNG sends OSLC (◆)

2. User thinks:

**! I need custom OSLC (◀) for my Table !**

3. User speaks:

JTS, tell DNG to send custom OSLC (◀)

RR, accept custom OSLC

4. Now, user says:

I have OSLC (◆) & custom OSLC (◀)

(a) OSLC-based communication

↓ ID	Name	↓ Specification	Min	Max
	Req			
450902	450902: Charge Controller Req	Charge Controller Req		
450902	450902: Charge Controller Req	Charge Controller Req		
450900	450900: System Req	System Req	1	2.5

(b) Modified table view based on OSLC

Figure 6.15: Behaviour of single RMM PA and RTC Client with multiple-workspaces

Minimum and maximum were requested to be, and were, transferred as “tags” to the model. The artifact of interest to the “User” is represented as the “green pentagon”. This is used to mirror OSLC’s description of resource shapes (see Section 3.8). The default information transferred is the represented as the “blue rhombus”, but a request was coded as shown in Figure 6.15(a) to request additional information, as desired.

The request put in by the “User” to JTS mentioned that, “Minimum” and “Maximum” shall be named “Min” and “Max”, respectively. The tag values were then used to populate the customised table view, as shown in Figure 6.15(b).

## 6.6 Summary

In this chapter, the systems and tools used in this work were abstracted as physical and logical structures. They were also described as separate PAs. This formed the basis on which relationships were established from each PA to others based on the underlying ALM technology. The tests for the solutions were consequently described as vendor-based, but as abstracted from the vendor-neutral standards and technologies they are based on. They were, however, adopted and tailored for use as is suitable for deployment in ASIC development processes. Although the implementation is vendor-specific, the solutions were first depicted in their existence as based on standards and technologies. Therefore, independent of any specific vendor. The particular tools used were then integrated as a tool-chain. Some properties of the tools were used as based on the underlying standards and technologies. The used properties were thus validated as being compliant with the enabling standards and technologies. Some of the ideas from Chapter 4 were applied and extended, to provide a suitable method for the traceability of ASIC requirements.

# 7 Case study-based evaluation

In Section 1.2, the research approach that was adopted at the start of this work was introduced. This research approach considers perspectival comparison of the resultant methodology to existing ones, as already discussed. This gives rise to the analysis and evaluation of the results of this work in comparison to existing theories and methodologies, as well as against available processes and standards. Based on this research approach, the developing methodology was applied to a development project. This was used to test the usefulness of the resulting methodologies in its application industrial to ASIC development.

The proposed methods were first applied to the architecture artifact development domain and the requirement database as a composite system. This was used to evaluate the resultant methodology for requirement consistency. Afterwards, other enabling systems of the ALM technology described in Sections 3 and 6 were applied for evaluation as an ALM-based methodology for ASIC development.

The methodologies were also evaluated for their usefulness in a collaborative development process. This involved nine (9) people, with each of them using any random number of multiple workspaces to test their adaptation in a collaborative team.

The procedure was applied to an ASIC pressure sensor currently under development. Figure 7.1 shows some of the components of the pressure sensor.

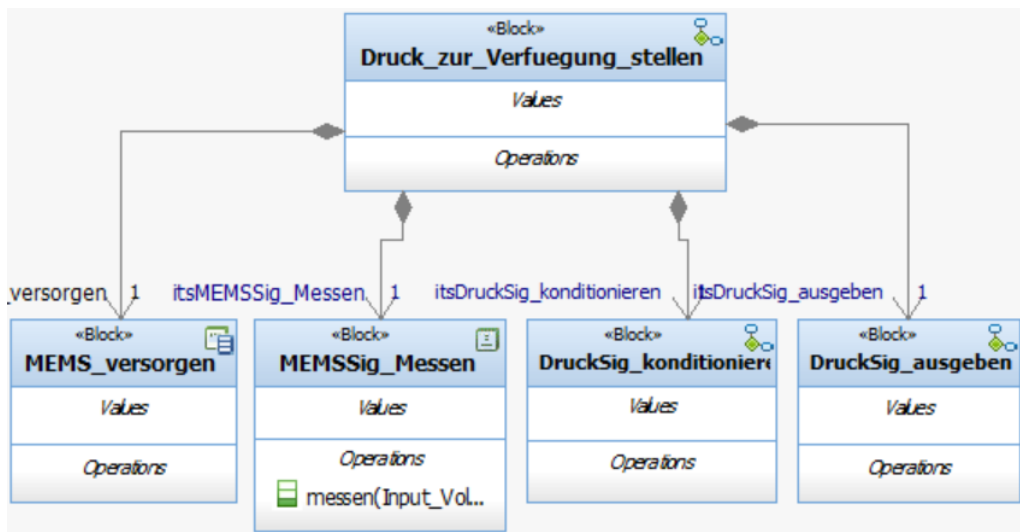


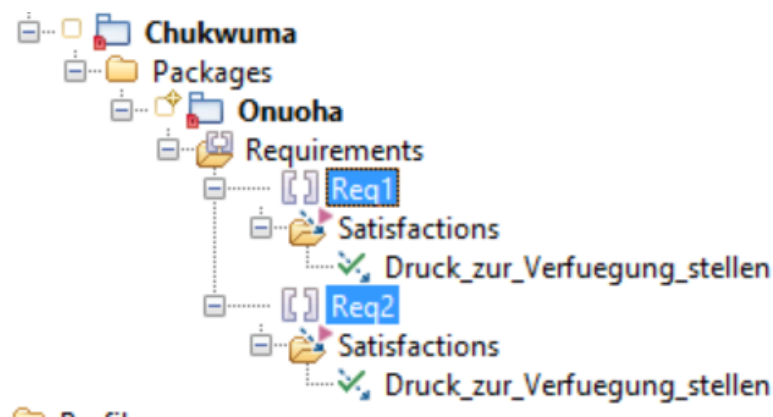
Figure 7.1: Default table view



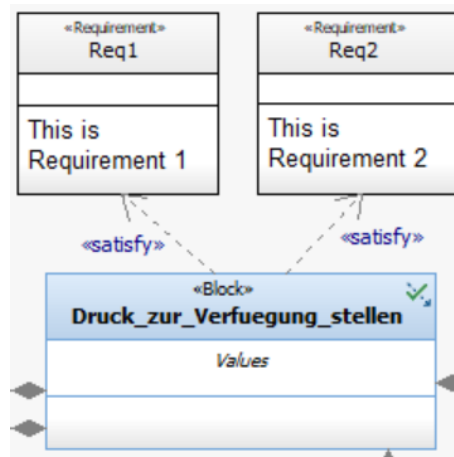
The image in Figure 7.1 is a block diagram of five blocks, which represent five functions of the ASIC pressure sensor. The “MEM\_versorgen”, “MEMSSig\_Messen”, “DruckSig\_konditionieren” and “DruckSig\_ausgeben” blocks are parts of the “Druck\_zur\_Verfuegung\_stellen” block. Together, they are all part of the “Provide Pressure” function of the pressure sensor.

## 7.1 Consistency of requirements

Requirements in SysML can either be defined as a modelling system-based or as a RM database artifacts. At the start of this research, requirements were created in the modelling domain. They are then associated with the models that fulfil them. The models and requirements are linked as shown in Figure 7.2.



(a) RR-based requirements



(b) Relationship with model

Figure 7.2: Method at the start of the research

“Req1” and “Req2” were defined and added as modelling system-based requirements to the FTR\_Project<sup>40</sup> model, as shown in Figure 7.2(a). The “Druck\_zur\_Verfuegung\_stellen” block was linked to these requirements, as captured in Figure 7.2(b). After the “satisfy” relationship was established between the model and requirement artifacts, the “Satisfactions” dependencies were automatically created for each of Req1 and Req2. This is captured in Figure 7.2(a).

In as much as the requirements were added and used as is standardised by SysML, these requirements only become visible to other collaborators after the new state of the model is checked in. That is, until this addition of requirements and their relationships to the model artifacts are checked in as updates, the updated model is not replicated across the collaboration platform.

In Sections 5.1.5 and 6, it was described that the PAs maintained under the same JTS have a common authentication credentials. This JTS feature was relied upon to provide RM PA access to the model in RR. When launched, RR accesses the model available in the sandbox. This model has the connection-link to the respective requirements available in DNG. Login was executed to the DNG PA from RR to establish the connection to the RM PA. Furthermore, using OSLC, creating requirements in DNG from the RR interface is possible. The requirements used in DNG are exported to RR using OSLC standards (see Section 3.8). These can be seen in Figure 7.3.

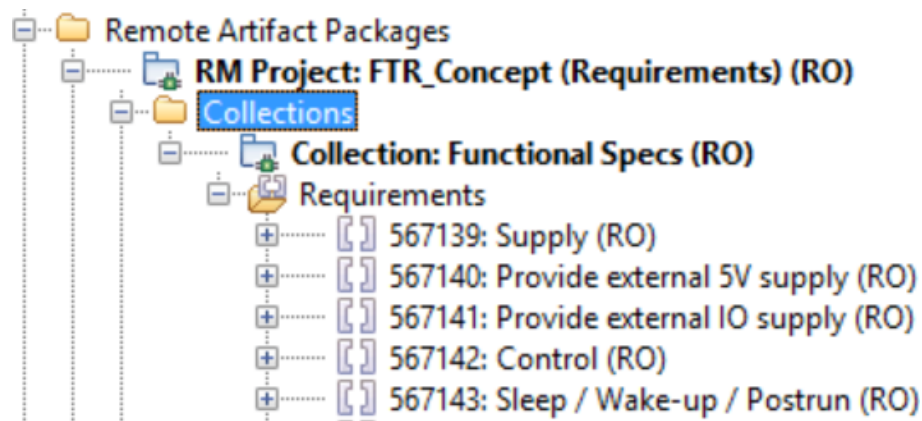


Figure 7.3: DNG-based requirements available in RR model

Figure 7.3 shows the RR view of some requirements for the FTR\_Project which are defined and located in DNG. In the figure, the RM PA shows “FTR\_Concept” as the “Requirements” used for the project. They are transferred to RR as a set of “Remote Artifact Packages” with “RO” (read only) attributes. Figure 7.3 also

<sup>40</sup>The model and the PAs used are named “FTR\_Concept”. For the model, the “Project” is appended. For the PAs, any one of “AM”, “CCM”, “RM” or “QM” is attached to the name, to show it is an Architecture, Change and Configuration, Quality, or Requirement Management PA.

shows a “Collection” as “Functional Specs”. Other collections can be added as part of the project. It is also seen that, each requirement is identified by its unique ID.

The requirements are transported to RR using OSLC standards. The image in Figure 7.4 below shows the same set of requirements in their DNG location.

ID ^	Name	Contents	Artifact Type
567139	Supply	2 Supply	Requirement
567140	Provide external 5V supply	3.1 Provide external 5V	Requirement
567141	Provide external IO supply	3.2 Provide external IO	Requirement
567142	Control	4 Control	Requirement
567143	Sleep / Wake-up /	4.1 Sleep / Wake-up /	Requirement

Figure 7.4: Sample DNG view of FTR\_Concept\_Project’s requirements

The requirements shown in Figure 7.4 are exact replicas of those in RR in Figure 7.3. They were defined in DNG and transported to RR via the OSLC connection. The attributes available in RR are dependent on those defined in DNG. As RO artifacts, any modification, additions, derivations or refinements were transported back to RR. One of such attributes is the IDs of the requirements. The IDs are unique across an entire JTS. Therefore, they cannot be duplicated. This means that, once used, it can only refer to a single requirement artifact. Any modification to that artifact is transported across the entire platform where it is used.

The requirements were associated with the model artifacts. Below is an illustration of using the requirements as model artifacts using Figure 7.5.

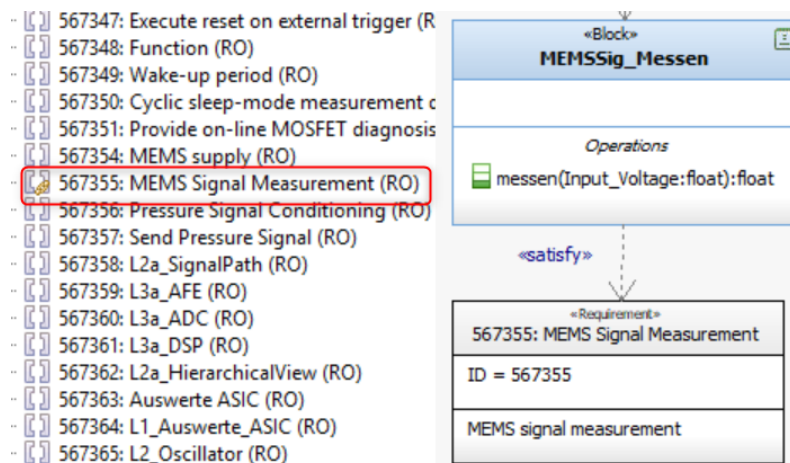
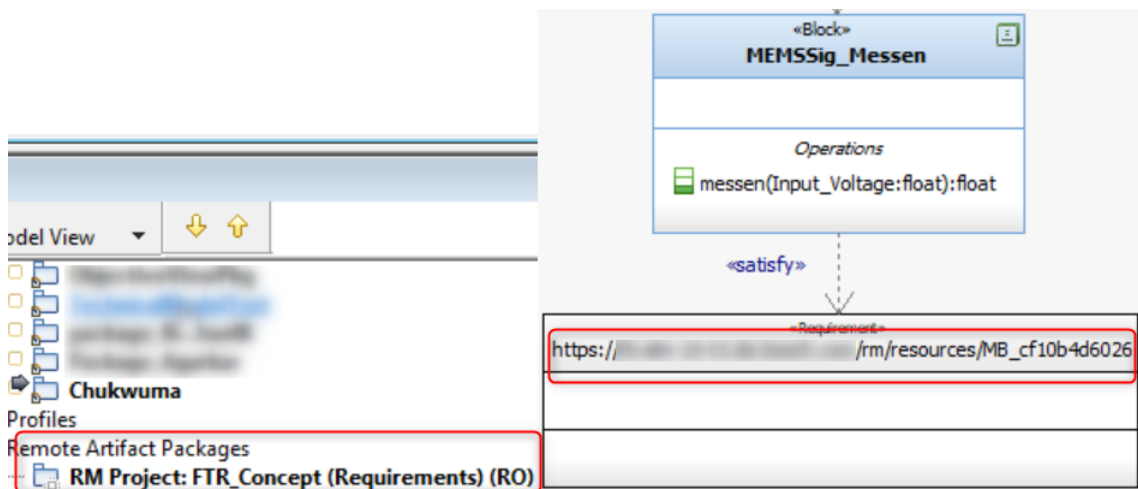


Figure 7.5: DNG-based requirements used in a block diagram

The “MEMSSig\_Messen” is one of the blocks introduced at the beginning of this chapter. The requirement “MEMS Signal Measurement” with ID “567355” associated with the MEMSSig\_Messen model artifact, as shown in Figure-7.5. The requirement assumes the attributes as they are. These attributes cannot be modified from the diagram, because they are RO.

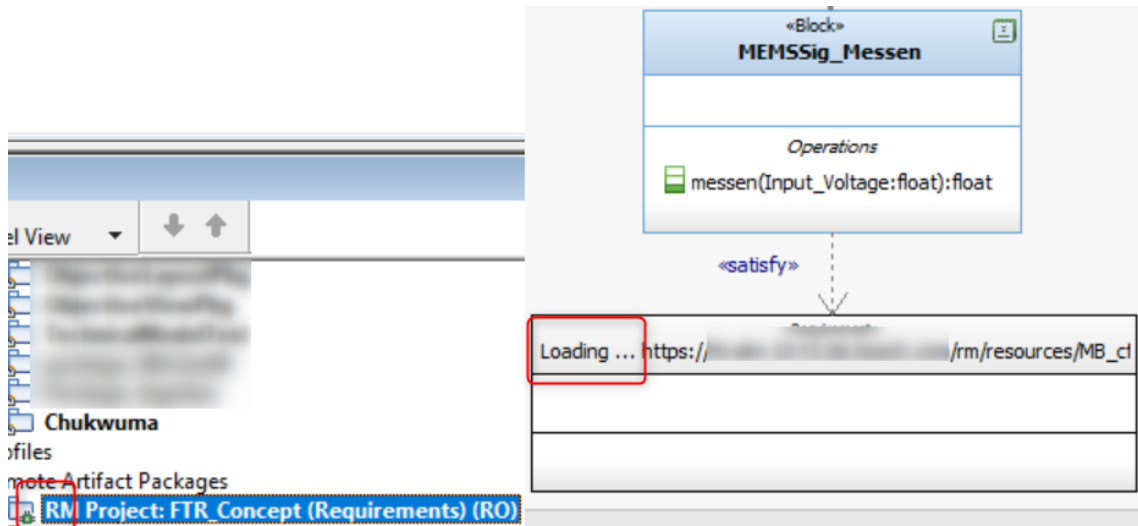
Unlike with the modelling system-defined requirements, any modifications to the database-defined requirements is visible to all the workspaces connected to the PA in real-time. As already mentioned, creation of requirements from RR is possible. This requirement addition is effected in the requirements database, but is done using RR as the modelling tool. Furthermore, all the requirements used from the RM PA collection are listed RR as remote artifacts. This is not the same with requirements defined in the tool for the modelling, where the requirements can be located anywhere on the model. Using the requirements as remote artifacts promotes a unified view and awareness of the requirements available for a project.

With the requirements located in DNG, login into DNG must be executed in order to access the requirements. Without logging in, the designed elements and parts of the system which depend on those requirements are still available and visible. However, the requirements are only visible as remote artifacts with internet links. This restricts the access the RMM-based model has to the requirements domain for the session. Figure 7.6 shows a summary of this scenario.



(a) Before login

Observe that the login has not been executed, as seen in the left hand side of Figure 7.6(a). Without the login, the relationship between the “MEMSSig\_Messen” block and the requirement is still visible from the right hand side of Figure 7.6(a). The “https://xxx” seen on the right hand side of Figure 7.6(a) indicates that the requirement is a remote artifact.



(b) Immediately after login

Figure 7.6: Remote requirements as internet links

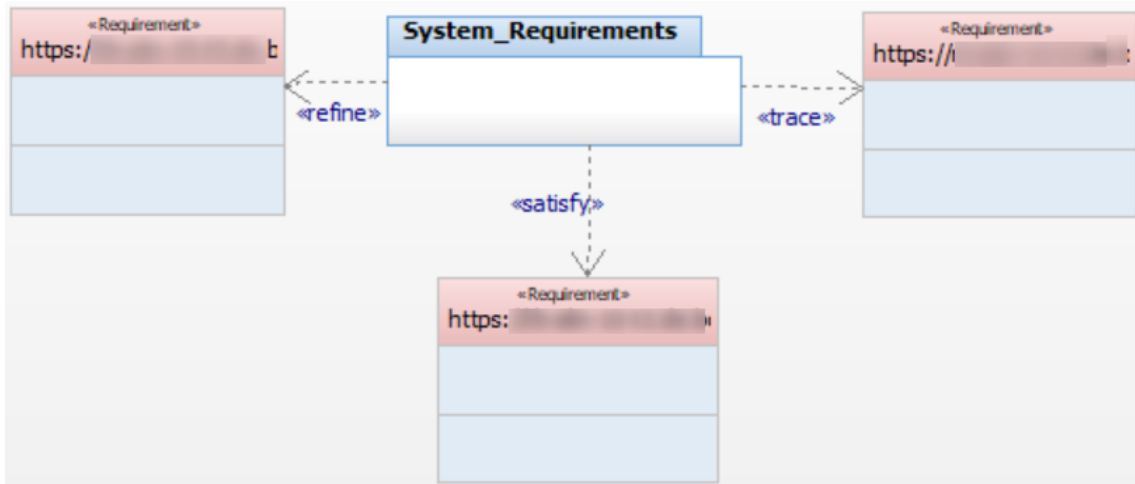
Figure 7.6(b) shows the same set of artifacts as Figure 7.6(a). However, the login has been executed as observed in the left hand side of Figure 7.6(b). This triggers the “Loading...” information visible on the right hand side. When fully loaded, the artifacts located in the DNG PA are available and the view of the block diagram becomes as shown in Figure 7.5.

### Summary of consistency of requirements

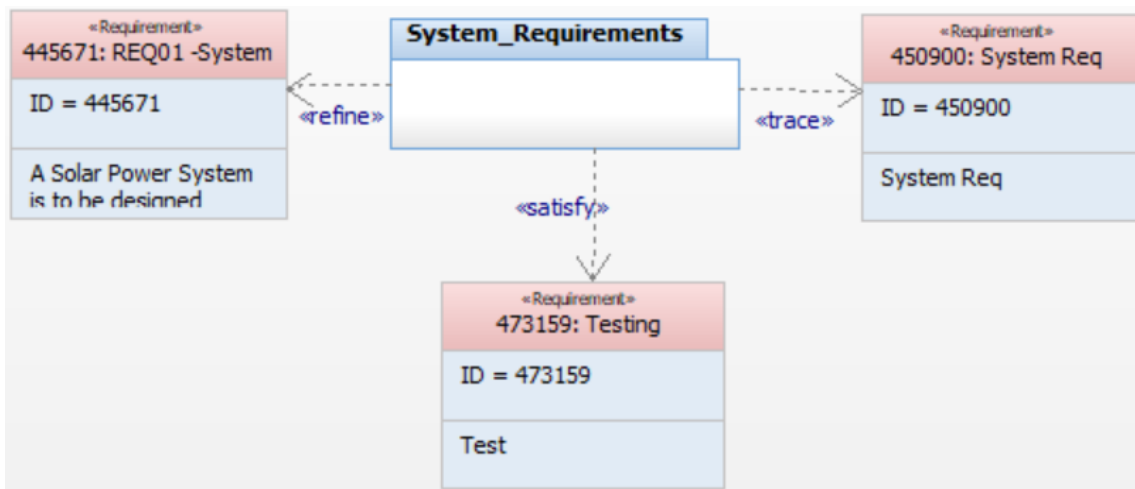
From the evaluations above, it was established that the use of RR-based requirements limits the collaboration possible amongst teams. Defining the requirements as RR-based creates conflicts and inconsistencies that are avoidable. This is because, while other members of the team would receive the updates and added requirements after the model is copied to RMM, the exact location of the requirement would not easily be known. The changes made to each requirement will also not be versionable, as each individual can make changes to the requirements concurrently. Furthermore, the requirements are not uniquely identified using RR-based requirement artifact.

In contrast, any addition or changes made to requirements in DNG are transported to all other PAs where this particular requirement is used. The IDs in DNG are also unique across the entire JTS area. This means no two different requirements can share the same ID in the entire JTS where your requirements are defined. Furthermore, only one person can make any modifications to a requirement at any point in time. This eliminates inconsistencies that might result from two people making different modifications to the same requirement concurrently. The chances of that happening in a DNG, is completely eliminated.

As stipulated by the OSLC standards from Section 3.8, clients can cache copies of system resources defined by servers as cacheable. This ensures that requirements already linked to model artifacts are available even without logging in to the requirements database. Thus, they can be used to populate requirement diagrams when needed, without the need for a log-in. If requirements are used in this manner, the internet addresses are resolved to the requirements they refer to immediately log-in is executed. This is illustrated using the requirement diagrams of Figure 7.7.



(a) Unresolved addresses before login



(b) Resolved addresses after login

Figure 7.7: Requirement diagrams with remote artifacts

Figure 7.7(a) shows a requirement diagram for “System\_Requirements” populated by remote requirements before a login was executed. These same requirements are visible again after login was executed. Once logged in, the internet addresses were

resolved to the actual names and IDs of the requirements available in DNG as shown in Figure 7.7(b).

The choice is not really between the use of a RM database and SysML tool for requirements definition for use in modelling. Rather, it is on how to use the former to replace the latter. The evaluations and comparisons done still highlights the advantages a RM database has over the use of model-defined requirement artifacts. Whereas the use of requirements defined in the model does not hold any advantage over the use of a RM database, where team collaboration is desired.

## 7.2 A working tool-chain

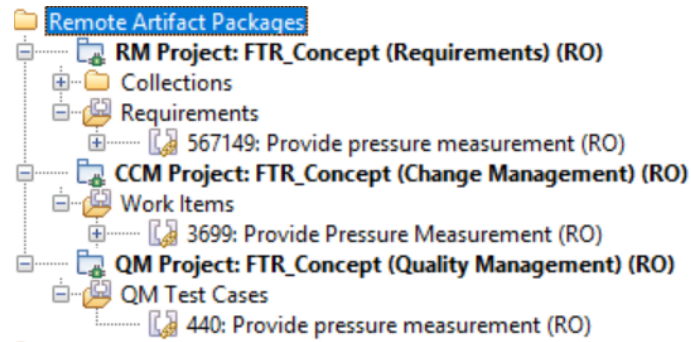
Other tools in the Rational platform were deployed, as discussed in Section 5. These tools have already been described as JTS platform enabled systems in Section 5.1. They were added to the FTR PA area as explained in Section 6. As the access and authentication is handled by the JTS, access to one is access to all.

When added, “Change Sets”, “Change Request” and “Requirements” belong to the “Uses” option under “Associations” in the RMM management console. The former two are related “Change Management”, while the later appear as “Requirements”. Another option under which the added PAs appeared is “Provides”, which relates to “Architecture Elements” under Associations, belonging to the added QM.

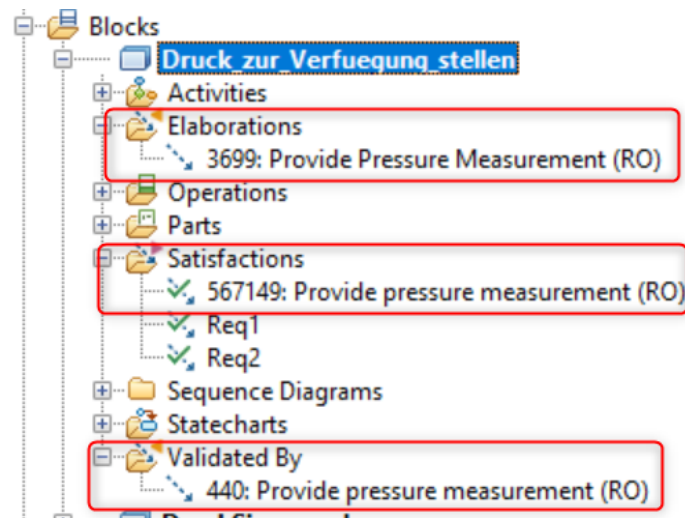
Thus, change sets and change requests were created in the CCM of RTC from the RMM PA and from RR. Sections 3.8 and 6.2 describe the technology-standard and implementation. Test cases were also created from within the RR. The created artifacts resided in the respective locations: requirements in DNG, change requests in CCM, and test cases in QM; irrespective of where they were created.

The essence of this exercise was to evaluate the capacity of the systems to work together as a collaborative platform. As OSLC deliverables, the trace and tracking of the artifact depended on the type and category of the requirements or work items (see Section 5.1.1). Each link to any of the artifacts used in this exercise was visible from the system that it was linked to. Model artifacts linked to CCM, DNG and QM were all bi-directionally visible: from RR, CCM and QM.

Additionally, the use of a naming convention that was described in Section 6.3 ensured an orderly tracking of the artifacts and their relationships. The requirements in DNG, the tasks required to implement them, and the test cases for verifying their implementation all carried the same or similar names. Where possible, the same name was assigned to the related requirements to make their relationship apparent, as shown in Figure 7.8.



(a) Remote artifact section in RR



(b) Associations to model artifact

Figure 7.8: Display of remote artifacts linked to RR models in RR

In the figure above, the “Remote Artifact Packages” comprises of the “RM”, “CCM” and “QM” PAs of the “FTR Project”, as captured in Figure 7.8(a). The displayed requirement ID is “567149”, while “3699” and “440” belong to the “Work Items” and “Test Cases” section of the remote artifacts. These are located in DNG, CCM and QM respectively. The association of the remote artifacts to the “Druck\_zur\_Verfuegung\_stellen” block is visible in Figure 7.8(b). The relationships of the block the remote artifacts that are associated with it is any one of “Elaborations”, “Satisfactions” or “Validated By”. Elaborations are CCM work items, while Satisfactions and Validations are DNG and QM artifacts respectively.

Comparing both images in Figure 7.8, it can be observed that the IDs and names of the remote artifacts match their corresponding IDs and names in the associations with the model artifact. It can also be observed that the remote artifacts all have similar names. This makes associating them to their model artifact easier<sup>41</sup>.

<sup>41</sup>The German phrase “Druck zur Verfuegung stellen” translates to “Provide pressure” in the



## Baselines, profiles and snapshots

Where modifications to a system are possible, it is also necessary to have the option of returning to previous versions of the system. This refers to the concept of SCM and VC discussed in Section 3.6.3. “Baseline” and “Snapshot” are the mechanisms for SCM and VC provided for the Rational platform.

The components were baselined, while snapshots were made for streams (see Section 5.1.4). These were used to revert to previous configurations and versions of the model. The baselines were also added to multiple PAs, while the snapshots were used to create further streams. These were used for collaboration with the nine (9) people mentioned at the beginning of this chapter.

A set of model artifacts developed by a team, but used by another in a development project can be added as a component. The component containing the artifacts to be protected is locked to avoid unauthorised modifications. The component can be provided for use as versionable artifact. This follows the principles described in Sections 5.1.4 and 6. Customised RR profiles were checked-in to the FTR Project in the RMM PA using a new component and the component was locked, as shown in Figure 7.9.

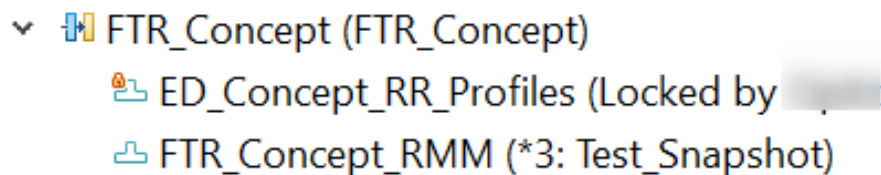


Figure 7.9: Access management using multiple components

“ED\_Concept\_RR\_Profiles” containing project-specific SysML profiles was “Locked”. Only the person that locked it can check-in new updates to the component. The profiles were added to the existing model available in “FTR\_Concept\_RMM” component. Thus, the “FTR\_Concept (FTR\_Concept)” stream contained two components, but updates from every stakeholder are only allowed into the one containing other model artifacts.

## Summary of the resultant tool-chain

RR, DNG, CCM and QM are the four tools used for creating and processing artifacts. Each tool is used to create its own standard artifact type, but each artifact type can be used by the other tools. It is therefore noteworthy that, although each tool can create those artifact type it consumes, it does not create them in its database. Rather, it requests the input fields from the database that owns the artifact type, and

---

context it is used

supplies the information needed to create the new artifact. Furthermore, multiple PAs can be used and associated with a PA. This ensures that existing artifacts are reusable where applicable.

The use of these four tools is enabled by the underlying systems and technologies. ALM and OSLC as technologies help deliver these capabilities. Using the four tools will help get the best out of the platform and tool-chain, delivered by the technologies. This is also going to depend on the implementation of these technologies. The evaluation here is strictly based on the capability delivered by CLM.

As a SCM and VC-based tool-chain, the baselines and snapshots created makes it possible to make copies of the configuration of the models as versions. These were used to create other PAs. That is, a PAs were created, and the available baselines and snapshots were used to initialise the newly-created PAs. This evaluates that back ups of PAs can be made available for future use or reference.

### **7.3 Evaluation of matrix views and tables**

The distribution of the system as a project platform for models to requirements traceability has been established. The different types of associations have been tested and the relationships have been visualised. To visualise the information carried by these associations and relationships, the model was viewed using Matrix Views and Tables that were described in Section 5.2.

#### **Matrix view**

Section 5.2.2 presented matrix views, and the how complex it can be when dealing with a model artifacts with many requirements. The FTR ASIC Project requirements are of such complexity. To extract the better performance with matrix, the method described in Section 6.5.3 was used and the results observed below.

Using this method, it was observed that matrix views can be generated for each package in the diagram, as well as different hierarchies of the packages. This method ensured the matrix view is populated with information without exceeding the screen capacity, as shown in Figure 7.10.

To: Requirement	From: Block	Scope: FunctionalModel
579583: Test RR Context 1	Druck_z...	MEMSSig...
567178: Security		
567152: Provide SPI Slave Interface		
567355: MEMS Signal Measurement		567355: MEMS Signal
567149: Provide pressure measurement	567149: Provide pressure	

Figure 7.10: Confirmation of remote DNG artifacts in a matrix view

Figure 7.10 shows a cross-section of the matrix created by scoping the matrix layout from “FunctionalModel” to “Requirement”, while setting the scope for the view to “FTR Concept” project. It can be observed that the requirements with IDs “567149” and “567355” which have been used for the “Druck\_zur\_Verfuegung\_stellen” and “MEMSSig\_Messen” in the previous sections are visible in the matrix, while other cells are empty. From the matrix view, any requirement that is supposed to link to a model artifact can simply be added on the matrix as an “OSLC Link”.

## Table view

Tables were described in Section 5.2.1 as being “more pragmatic”. Its implementation has already been described in Section 6.5.3. The extension applied to tables was also discussed and the result shown in Figure 6.15(b).

Project-specific artifacts were created and required attributes were applied. These artifacts were added to the column section of DNG (see Table 6.2). To analyse this result using the FTR Project, the images shown in Figure 7.11 are used. They compare the output of the default “Remote Artifacts” and the extension applied to a table in the FTR Project.

ID▲	Name	Implemented By ◀	Maxim...	Minimum	ASIL
567149	Provide pressure measurement	3699: Provide Pressure Measurement	1.000000	2.000000	B

(a) Context view in DNG

ID	Name	Specification	Link Type	Link From
567149	567149: Provide pressure measurement		Satisfaction	Druck_zur_Verfuegung_stellen

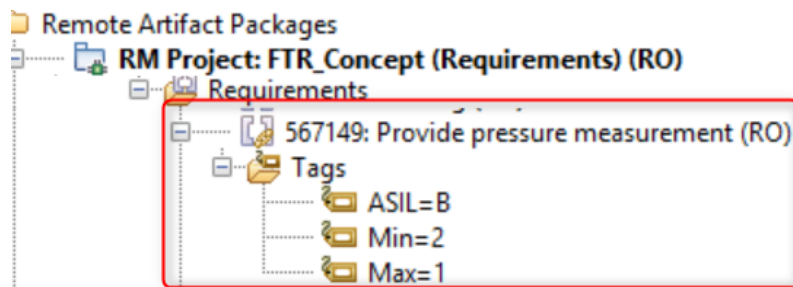
(b) Default table in RR

```

"tags": {
  "Min": {
    "property": "Minimum"
  },
  "Max": {
    "property": "Maximum"
  },
  "ASIL": {
    "property": "ASIL"
  }
}

```

(c) Applied JSON code



(d) Remote artifacts section in RR

Name	Min	Max	ASIL
567149: Provide pressure measurement	2	1	B

(e) Customised table

Figure 7.11: Display of available requirement artifacts

In Figure 7.11(a), “Maximum”, “Minimum” and “ASIL”<sup>42</sup> were defined and added to the DNG column as custom artifacts. The attributes were defined to use the data type “Float” and “Enumerated” as appropriate (see Table 6.1). These custom artifacts were not automatically transferred to the RR table shown in Figure 7.11(b). Neither were they visible in the remote artifacts section of the RR model view (see Section 7.1). The requirement attributes sent to RMM by default are the five information fields shown in Figure 7.11(b).

Figure 7.11(c) is a snippet of the JSON code that added “tags” as a request for the information sent from DNG. This code was added to configuration file mentioned in Section 6.5.3. The code requests that, the DNG properties, which are the named attributes available in DNG, be sent to RR (see Section 6.5.3). In Figure 7.11(d), “ASIL”, “Min” and “Max” are attached as “Tags” to the requirement with ID “567149”. The same requirement is visible in the RR remote requirements table in Figure 7.11(a) in DNG, as well as in the RR table of Figure 7.11(b).

The Tags returned were selected as Table Layout information. The resultant table is as shown in Figure 7.11(e). Both table views of Figure 7.11 are available in the RR model. However, this information is not automatically propagated to RMM. Furthermore, this method only configures the View for each workstation or for each project.

The entries made in DNG do not go through the change sets. Therefore, “Accept” action was not required to transport the updates to RR. Rather, the image in Figure 7.12 captured the appearance of the RMM connection indicator.

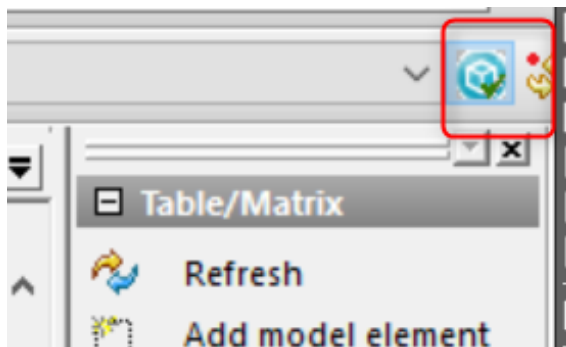


Figure 7.12: Notification for obsolete requirement artifacts

Figure 7.12 shows the view of RMM-based model when the information in DNG and that which the model currently has are not synchronised. The red dot indicates that a refresh is required. The top right hand side of Figure 7.12 is the RMM refresh button. Refreshing the model or just the RMM connection, followed by a refresh of

<sup>42</sup>ASIL is an acronym for Automotive Safety Integrity Level, a risk classification metrics defined by the ISO 26262 standards.

the table ensures the updated values from DNG were retrieved. This information populates the table automatically. This concept has already been described (see the illustrations of Figures 6.2 and 6.7).

### **Gap analysis and tracing**

The main focus of this work was born out of the need for easy and efficient traceability. Providing a concise and compact view of the relationships between the different artifacts is one efficient way of achieving that aim. A concise and compact view has already been established using matrix and table views. The use of such methods as matrices and tables provide what is known as “Gap Analysis”.

Gap analysis is any means of analysing the model artifacts and matching them against the requirements that link to them. That is, a model artifact can be profiled and matched with the requirements. Alternatively, a requirement artifact is profiled and matched against model artifacts. This analysis is more effectively done using tables and matrices. When matched, any cell that is empty in the matrix or table shows that a trace is missing. Also, the requirements can be visually compared to the models to find any missing links. In the event of a missing link during the analysis, the stakeholder responsible for fulfilling the missing part can be tagged for notification.

Matrices and tables provide comprehensive views of the model and requirement artifacts’ relationships. It ensures that visual traceability between the model and requirement artifacts can be established using the links provided by the different systems (DNG, RMM, RR and RTC).

Gap analysis was performed using matrix and table view methods. Missing items were added directly on the matrix views and tables. Some modifications were also made and the newly added information were propagated through the collaborative platform. Every workspace connected to the FTR Project received the updated information when they were checked-in.

## **7.4 Recommendations**

To promote uniformity in the usage of the prescribed methods, a uniform approach by all collaborators on a project using the tool-chain is required. Recommendations were applied for each workspace associated with the FTR Concept Project. It is premised as an action and decision mechanism for promoting uniformity in the practice of MBSE. It was based on the evaluations made on its application to the FTR Concept Project. These recommendations consider the basic setup according to the preceding sections as already in place. This approach is specified by the use of the flow charts, as shown in Figure 7.13.

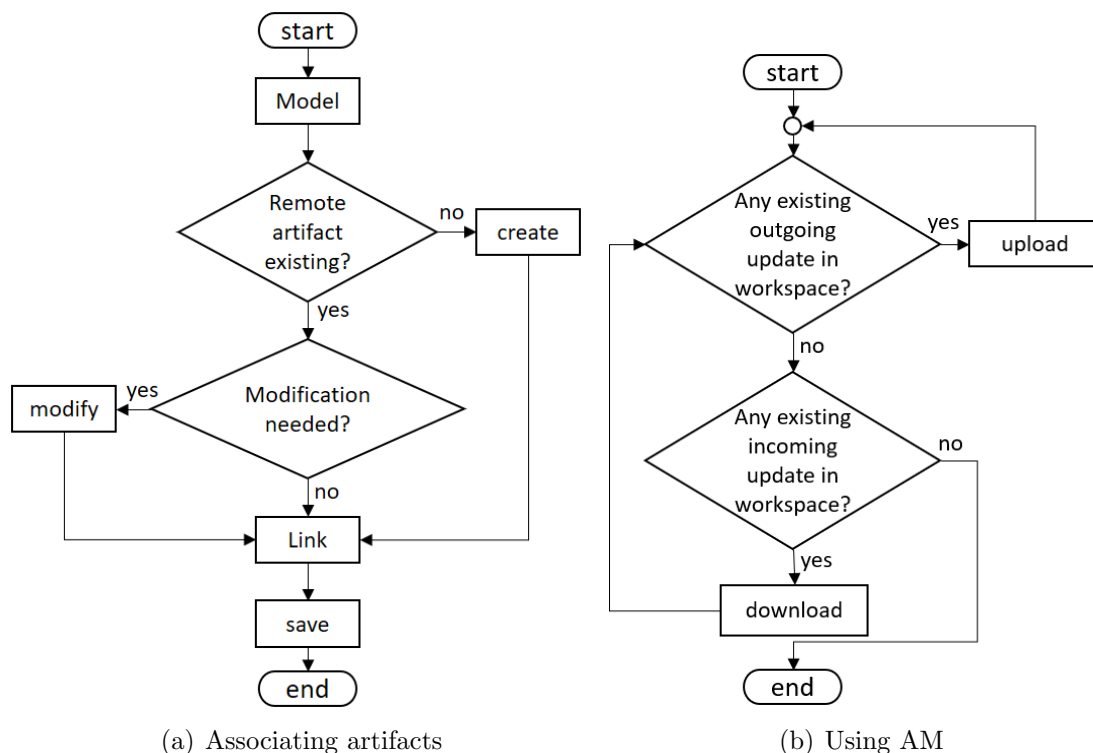


Figure 7.13: Decision process for using the platform

This is particularly with respect to the module developers and the requirements they fulfil with the models. Figure 7.13(a) proposes a decision-based process for the linking of artifacts. It starts with a model artifact in RR. If the addition or modification of the artifact in RR is complete, it is linked to other artifacts of interest. The artifact could be a requirement or a test case. It can also be linked to a work item addressed to a collaborator. If the required artifact to be linked to the model is not available, it is first created. If it is existing, but incomplete, it is modified. The model is saved afterwards. After saving the model, it is checked in to the associated workspace. This makes it available as a server-based artifact.

The next part refers to the download and upload of the model to the AM PA. To avoid conflicts between different versions of collaborative artifacts in the participating workspaces and loss of data, the process captured in Figure 7.13(b) is recommended. Before checking in to the RMM of the specific project, it is recommended to first check for available updates in the workspace. This helps to keep the flow of the data in and out of the workspace sequential. Although updates available in a workspace can be ignored before existing the collaborative platform, it is recommended to retrieve the available changes before exiting. This helps to keep the sandbox associated with the workspace current and consistent with the latest available collaboration information.

## 7.5 Summary

This chapter focused on applying the methodologies and procedures that resulted from the previous chapter. It highlighted, referenced and exemplified the inconsistencies that are introduced by the use of model-defined requirement artifacts. This was immediately contrasted against the use of a RM database. The model and requirement artifacts were thus linked. Following the evaluation of remotely-defined requirement artifacts was the association of the other PAs and their peculiar artifacts to the already established relationship between model and requirement artifacts. The analyses of using names to effect an easy traceability approach amongst the different sets of artifacts were evaluated. Configuration, security and protection measures for access restriction to specific artifacts were also evaluated. This included application of baselines and versioning of the artifacts. This was followed by the analyses of using extended and customised tables and matrices for gap analyses and tracing amongst the artifacts. The answers to some of the questions from Section 1.1 were explicitly and implicitly provided. Recommendations were made for the use of the resulting tool-chain to promote uniformity of use across the development platform as a sequence of actions and decisions. These recommendations



# 8 Conclusion

## 8.1 Summary

As stated in the research objectives of this work (Section 1.1.2), the focal point of this research work is on the development of actionable steps to observe in order to promote MBSE in ASIC development. Part of these steps include the effective traceability of model artifacts to the requirements they fulfil. The steps also evolve into a set of methodologies.

To understand what methodologies entail and to provide them, the work started with a research on methodologies. The applied approach in this work is documented in Section 1.2. Based on this applied approach, some questions were developed as a guide towards the realisation of a required solution. These questions were introduced in Sections 1.1.1 and 1.1.2 as the research questions. They did not seek to find desired answers, but were focused on realising outcomes that will guide the decision makers on what course of actions to take going forward. For reference purpose, these questions are repeated here.

Under “Aim and scope of this thesis” (Section 1.1.1), the questions were:

1. Is requirements traceability in ASIC module development possible?
2. Does it improve communication among the stakeholders?
3. Can it help provide the missing contexts of requirements?
4. Does it improve efficiency?
5. Will it spawn a more pragmatic development process?
6. Will the process be better managed?

The questions above were developed from the informal “kick-off” meetings and interview sessions that were conducted in the discovery and knowledge acquisition phases of this research<sup>43</sup>. These sessions were with some people who are directly involved in the development of ASIC in some way. People who have direct impact in the ASIC development have been introduced as “stakeholders” in Chapter 1. Each stakeholder faces different challenges in the development process.

---

<sup>43</sup>The knowledge of the stakeholders of the ASIC development process were collated

To provide answers to the above research questions, the following summary of questions were developed as “Research objectives” in Section 1.1.2:

1. What are requirements?
2. What is requirements traceability?
3. Is traceability specific to a particular systems’ domain?
4. Are there standard practices?
5. If yes, how can other solutions be improved or adapted?
6. How can it be used in ASIC development?

This approach was formed in line with the hybrid, solution-oriented research analysis methodology deduced from Section 1.2. The idea is to answer the first set of questions by providing answers to the second. An answer might be objective or subjective. The answers are not necessarily chronologically aligned with the questions. In this chapter, the answers are chronicled and cited as applicable. For context, the first sets of questions are referred to as “**A**”, while the second set are tagged “**B**”<sup>44</sup>.

Starting with B.1 and B.2, the answers to “What are requirements?” and “What is requirements traceability?” can be found explicitly in Section 2.2 and 3.9. The discussions around those sections also provide the proper contexts for the answers. Those two answers lead directly to the answer of A.1 (“Is requirements traceability in ASIC module development possible?”). The answer to which is, “yes”, but depending on whether the module part can be developed using SysML. The answer is yes, because it has been used during the course of this work to test with sample parts being developed. However, if the part cannot be developed with SysML, but by a different tool, then an integration of the tool with SysML has to be studied. SysML is suitable for handling requirements as a part of a model and should be explored as an integrated solution with other modelling standards and systems. It should be noted that, Albinet et al [8] has worked on such approach with success.

Continuing with B.3 and B.4 as a single block of questions: yes, there are standard practices, relative to specific domains. These can be found in this work in Sections 3.9, 5 and 7, as well as in Delligatti [53:7]. There are also methodologies that are common to almost all. The use of tables, lists and matrices is common amongst all systems domains. Therefore, traceability of requirements is not peculiar to ASIC, in reference to B.4. If the ideas communicated in this work are judiciously adhered to, the framework provided by the implemented solutions are optimal for questions A.3, A.4 and B.5. The ideas include linking the additions made to their sources and destinations, as well as checking-in and accepting modified artifacts regularly. This is on the assumption that the requisite internet connectivity is available.

---

<sup>44</sup>“Aim and scope of this thesis” questions are tagged “A.1” to “A.6”, while those of “Research objectives” are “B.1 to B.6”.

Further analysis of Sections 5 and 7 partly reveals some suitable answers to question A.5. The complete answer is quite very subjective. However, the focus of this work is to provide answers and independent on whether the provided answer is favourable or not. The execution and manner of use of the provided methodologies will very much determine how well a pragmatic process ensues.

The pragmatism of the methodologies outlined in this work stems from the fact that, ALM technology and the implementation in CLM platform enhances collaboration. SE practice using ALM is advised, so as to promote efficiency. Which leads directly to the answer of A.6: yes, the use of the outlined methods and systems will provide a more involving ASIC development process, ensured by using a collaborative tool-chain. Improved collaboration ensures better productivity.

And lastly, the entirety of this work is the “how it can be used in ASIC production” (B.6), to the best of the availability of the parts of the modules to be designed, which is also related to the answer provided for B.4. Recommendations for the actual development of modules and their traceability to the requirements they fulfil is outlined in Section 7.4.

## 8.2 Future research

In Section 3.1 the idea of using models as FS was introduced. It also skimly mentioned that, specifications should be represented either as model or text artifacts, but not both. This would improve the efficiency of the development process. It would also promote artifact reuse and adaptation. While the use of model artifacts will promote more MBSE, some things might still be represented as texts. An example could be the contexts of a requirement or use case, as provided by the customer or as captured by the architect.

As is, this work focused on tracing the relationships between model and requirements artifacts. Traceability and relationship methodologies has been fulfilled by this work, having provided a method for linking and synchronising the model and requirements artifacts. It has also prescribed methods for visualising the relationships by the use of “views” and “gap analysis”. The next phase could be to separate the modelling and requirements domains as two different sets of specification documentation, but without repetitions. That is, the model-based and the text-based artifact specifications will be handled and processed in the AM and RM systems, respectively. This would require some automation and standard mapping methodologies.

Closely related to the future scope highlighted above is the merging of both domains of specification into a single documentation, especially one provided to the customer as user specification (see Figure 3.1). As at the time of this work, the documents related to the models and specifications can only be generated for each system separately. That is, documents are generated for the models from the SysML-based

tool. A different set of documents are also generated for the RM system. Generating a single document that will properly map and match the models to the requirements contextually and hierarchically will further enhance productivity.

The methodologies that have been prescribed here largely relates to implementing the ASIC modules. To evolve into MBSE, the module development will be based on SysML. The testing and verification of these modules is recommended to also adopt a MBSE approach. Only mention to testing activities in this work were related to the use of “mock tests” in order to naming conventions. However, certain tests, verification and validation activities are specific to the particular design and development process. This means that, adopting the methods prescribed in this work will likely render some of the other processes that depend on the already available processes unsuitable or inefficient to the newer methods. For this reason, studies that will focus on other processes that depend on the development process directly modified by this process are recommended. This would help review and streamline the interwoven methods into a development process that integrates all the design and development stages of the ASIC development.

# References

- [1] Quality Management, Statistics and Certification Standards Committee, *Project management - Project management systems - Part 5: Concepts*, Jan. 2009.
- [2] M. Luisa, F. Mariangela, and N. I. Pierluigi, “Market research for requirements analysis using linguistic tools,” *Requirements Engineering*, vol. 9, no. 1, pp. 40–56, Feb. 1, 2004, ISSN: 0947-3602, 1432-010X. DOI: 10.1007/s00766-003-0179-8. [Online]. Available: <http://link.springer.com/10.1007/s00766-003-0179-8> (visited on 02/11/2020).
- [3] Aberdeen Group, “The transition from 2D drafting to 3D modelling benchmark report: Improving engineering efficiency,” *Aberdeen Group, Inc.*, p. 25, September Sep. 2006. [Online]. Available: [http://images.autodesk.com/apac\\_korea\\_main/files/digital\\_prototyping\\_benchmark\\_report0.pdf](http://images.autodesk.com/apac_korea_main/files/digital_prototyping_benchmark_report0.pdf) (visited on 02/14/2020).
- [4] H. J.M. Veendrick, *Nanometer CMOS ICs From Basics to ASICs*, Second Edition. 2017. DOI: 10.1007/978-3-319-47597-4.
- [5] G. Liebel, M. Tichy, E. Knauss, O. Ljungkrantz, and G. Stieglbauer, “Organisation and communication problems in automotive requirements engineering,” *Requirements Eng*, vol. 23, no. 1, pp. 145–167, Mar. 2018, ISSN: 0947-3602, 1432-010X. DOI: 10.1007/s00766-016-0261-7. [Online]. Available: <http://link.springer.com/10.1007/s00766-016-0261-7> (visited on 02/26/2020).
- [6] C. Ebert and C. Jones, “Embedded Software: Facts, Figures, and Future,” *Computer*, vol. 42, no. 4, pp. 42–52, Apr. 2009, Conference Name: Computer, ISSN: 1558-0814. DOI: 10.1109/MC.2009.118.
- [7] J. Zimmermann, S. Stattelmann, A. Viehl, O. Bringmann, and W. Rosenstiel, “Model-driven virtual prototyping for real-time simulation of distributed embedded systems,” in *7th IEEE International Symposium on Industrial Embedded Systems (SIES'12)*, Karlsruhe, Germany: IEEE, Jun. 2012, pp. 201–210, ISBN: 978-1-4673-2684-1 978-1-4673-2685-8 978-1-4673-2683-4. DOI: 10.1109/SIES.2012.6356586. [Online]. Available: <http://ieeexplore.ieee.org/document/6356586/> (visited on 02/27/2020).
- [8] A. Albinet, J.-L. Boulanger, H. Dubois, M.-A. Peraldi-Frati, Y. Sorel, and Q.-D. Van, “Model-based methodology for requirements traceability in embedded systems,” p. 11,

- [9] R. Wieringa and A. Morah, “Technical action research as a validation method in information systems design science,” in *Design science research in information systems: advances in theory and practice: 7th international conference, DESRIST 2012, Las Vegas, NV, USA, May 14-15, 2012: proceedings*, K. Peffers, M. Rothenberger, and W. Kuechler, Eds., ser. Lecture notes in computer science, OCLC: ocn785082591, vol. 7286, Berlin, Heidelberg: Springer, 2012, pp. 220–238, ISBN: 978-3-642-29862-2. DOI: 10.1007/978-3-642-29863-9. [Online]. Available: <http://link.springer.com/10.1007/978-3-642-29863-9>.
- [10] H. Lempinen, M. Rossi, and V. K. Tuunainen, “Integrating organisational design with it design: The queensland health payroll case,” in *Design science research in information systems: advances in theory and practice: 7th international conference, DESRIST 2012, Las Vegas, NV, USA, May 14-15, 2012: proceedings*, K. Peffers, M. Rothenberger, and W. Kuechler, Eds., ser. Lecture notes in computer science, OCLC: ocn785082591, vol. 7286, Berlin, Heidelberg: Springer, 2012, pp. 52–65, ISBN: 978-3-642-29862-2. DOI: 10.1007/978-3-642-29863-9. [Online]. Available: <http://link.springer.com/10.1007/978-3-642-29863-9>.
- [11] M. K. Sein, O. Henfridsson, S. Purao, M. Rossi, and R. Lindgren, “Action design research,” *MIS Quarterly*, vol. 35, no. 1, pp. 37–56, Mar. 2011, ISSN: 02767783. DOI: 10.2307/23043488. [Online]. Available: <https://www.jstor.org/stable/10.2307/23043488> (visited on 02/22/2020).
- [12] E. Tüzün, B. Tekinerdogan, Y. Macit, and K. İnce, “Adopting integrated application lifecycle management within a large-scale software company: An action research approach,” *Journal of Systems and Software*, vol. 149, pp. 63–82, Mar. 2019, tex.publisher: Elsevier Inc., ISSN: 01641212. DOI: 10.1016/j.jss.2018.11.021.
- [13] D. Amalfitano, V. De Simone, S. Scala, and A. R. Fasolino, “A model-driven engineering approach for supporting questionnaire-based gap analysis processes through application lifecycle management systems,” *Software Quality Journal*, Jan. 17, 2020, ISSN: 0963-9314, 1573-1367. DOI: 10.1007/s11219-019-09479-w. [Online]. Available: <http://link.springer.com/10.1007/s11219-019-09479-w> (visited on 02/19/2020).
- [14] W. Hardt and W. Rosenstiel, “Prototyping of tightly coupled hardware/software-systems,” p. 35, 1997.
- [15] H.-J. Eikerling, W. Hardt, J. Gerlach, and W. Rosenstiel, “A methodology for rapid analysis and optimization of embedded systems,” in *Proceedings IEEE Symposium and Workshop on Engineering of Computer-Based Systems*, ISSN: null, Mar. 1996, pp. 252–259. DOI: 10.1109/ECBS.1996.494536.

- [16] T. U. Chemnitz. (Sep. 27, 2019). Research | computer engineering | department of computer science | TU chemnitz. Archive Location: Worldwide Last Modified: 2019-09-27 Library Catalog: [www.tu-chemnitz.de](http://www.tu-chemnitz.de), [Online]. Available: <https://www.tu-chemnitz.de/informatik/ce/research/research.php.en> (visited on 04/29/2020).
- [17] T. U. Chemnitz. (Sep. 26, 2019). Technische Informatik | Fakultät für Informatik | TU Chemnitz. Archive Location: Worldwide Last Modified: 2019-09-26 Library Catalog: [www.tu-chemnitz.de](http://www.tu-chemnitz.de), [Online]. Available: <https://www.tu-chemnitz.de/informatik/ce/research/areiom-adm.php.en> (visited on 04/29/2020).
- [18] N. Englisch, O. Khan, R. Mittag, F. Hänchen, A. Heller, and W. Hardt, “YellowCar: Automotive multi-ECU demonstrator platform,” 2017, ISBN: 9783885796695 Publisher: Gesellschaft für Informatik, Bonn, ISSN: 1617-5468. DOI: 10.18420/IN2017\_151. [Online]. Available: <https://dl.gi.de/handle/20.500.12116/3916> (visited on 03/04/2020).
- [19] T. U. Chemnitz. (Jan. 22, 2018). Technische Informatik | Fakultät für Informatik | TU Chemnitz. Archive Location: Worldwide Last Modified: 2018-01-22 Library Catalog: [www.tu-chemnitz.de](http://www.tu-chemnitz.de), [Online]. Available: <https://www.tu-chemnitz.de/informatik/ce/research/yellowcar.php> (visited on 04/29/2020).
- [20] T. U. Chemnitz. (Oct. 30, 2019). Teaching | computer engineering | department of computer science | TU chemnitz. Archive Location: Worldwide Last Modified: 2019-10-30 Library Catalog: [www.tu-chemnitz.de](http://www.tu-chemnitz.de), [Online]. Available: <https://www.tu-chemnitz.de/informatik/ce/lectures/lectures.php.en> (visited on 04/29/2020).
- [21] T. U. Chemnitz. (Jan. 5, 2016). Publikationen | Technische Informatik | Fakultät für Informatik | TU Chemnitz. Archive Location: Worldwide Last Modified: 2016-01-05 Library Catalog: [www.tu-chemnitz.de](http://www.tu-chemnitz.de), [Online]. Available: <https://www.tu-chemnitz.de/informatik/ce/publications/?controller=index&PersonID1=30> (visited on 04/29/2020).
- [22] BOSCH Automotive Electronics. (2019). About us, Bosch Semiconductors, [Online]. Available: <https://www.bosch-semiconductors.com/about-us/> (visited on 02/15/2020).
- [23] Bosch. (2020). The bosch group at a glance, Bosch Global. Library Catalog: [www.bosch.com](http://www.bosch.com), [Online]. Available: <https://www.bosch.com/company/our-figures/> (visited on 04/29/2020).
- [24] I Wagner. (Apr. 27, 2020). Top automotive suppliers 2018, Statista. Library Catalog: [www.statista.com](http://www.statista.com), [Online]. Available: <https://www.statista.com/statistics/199703/10-leading-global-automotive-original-equipment-suppliers/> (visited on 04/29/2020).

- [25] L. Trego. (Jun. 13, 2019). Infineon acquires cypress, AUTONOMOUS VEHICLE TECHNOLOGY. Library Catalog: [www.autonomousvehicletech.com](http://www.autonomousvehicletech.com), [Online]. Available: <https://www.autonomousvehicletech.com/articles/1812-infineon-acquires-cypress> (visited on 04/29/2020).
- [26] A. Holst. (Jan. 9, 2019). Semiconductor demand worldwide by end use 2018, Statista, [Online]. Available: <https://www.statista.com/statistics/894267/semiconductor-market-share-worldwide-by-end-use/> (visited on 02/13/2020).
- [27] H. S. Bennett, “Will future measurement needs of the semiconductor industry be met?” *J Res Natl Inst Stand Technol*, vol. 112, no. 1, pp. 25–38, 2007, ISSN: 1044-677X. DOI: 10.6028/jres.112.002. [Online]. Available: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4654602/> (visited on 02/15/2020).
- [28] P. Kassanos, H. Ip, and G. Z. Yang, “Ultra-low power application-specific integrated circuits for sensing,” in *Implantable sensors and systems - from theory to practice*, New York, NY: Springer Berlin Heidelberg, 2018, pp. 281–437, ISBN: 978-3-319-69747-5.
- [29] BOSCH. (Nov. 5, 2018). Semiconductors – market of the future: Bosch is growing faster than the market, Bosch Media Service, [Online]. Available: <https://www.bosch-presse.de/pressportal/de/en/semiconductors-%E2%80%93-market-of-the-future-bosch-is-growing-faster-than-the-market-174464.html> (visited on 02/15/2020).
- [30] P. Valdes-Dapena. (May 20, 2015). How airbags (should) work, CNNMoney, [Online]. Available: <https://money.cnn.com/2015/05/20/autos/how-airbags-work/index.html> (visited on 02/15/2020).
- [31] D. Glassbrenner and M. Starnes, “Lives saved calculations for seat belts and frontal air bags,” National Highway Traffic Safety Administration, Tech. Rep., 2009. DOI: 10.13140/RG.2.2.15177.44647. [Online]. Available: <http://rgdoi.net/10.13140/RG.2.2.15177.44647> (visited on 02/15/2020).
- [32] P. Braun, M. Broy, F. Houdek, M. Kirchmayr, M. Müller, B. Penzenstadler, K. Pohl, and T. Weyer, “Guiding requirements engineering for software-intensive embedded systems in the automotive industry,” *Comput Sci Res Dev*, vol. 29, no. 1, pp. 21–43, Feb. 1, 2014, ISSN: 1865-2042. DOI: 10.1007/s00450-010-0136-y. [Online]. Available: <https://doi.org/10.1007/s00450-010-0136-y> (visited on 02/06/2020).
- [33] P. E. Lanigan, S. Kavulya, P. Narasimhan, T. E. Fuhrman, and M. A. Salman, “Diagnosis in automotive systems: A survey,” *Carnegie Mellon University*, p. 22, General Motors Research & Development Jun. 2011. [Online]. Available: <https://www.pdl.cmu.edu/ftp/ProblemDiagnosis/CMU-PDL-11-110.pdf> (visited on 03/04/2020).



- [34] D. H. Kim and S. K. Lim, “Impact of TSV and Device Scaling on the Quality of 3D ICs,” in *More than Moore Technologies for Next Generation Computer Design*, R. O. Topaloglu, Ed. New York, NY: Springer New York, 2015, pp. 1–22, ISBN: 978-1-4939-2162-1 978-1-4939-2163-8. DOI: 10.1007/978-1-4939-2163-8\_1. [Online]. Available: <http://link.springer.com/10.1007/978-1-4939-2163-8> (visited on 02/16/2020).
- [35] W. J. Spencer and C. L. Seitz, “Engines of Progress: Semiconductor Technology Trends and Issues,” in *Defining a Decade: Envisioning CSTB’s Second 10 Years*. Washington, D.C.: National Academies Press, Sep. 9, 1997, pp. 22–35, ISBN: 978-0-309-05933-6. DOI: 10.17226/5903. [Online]. Available: <http://www.nap.edu/catalog/5903> (visited on 02/13/2020).
- [36] A. Kirchner, J.-H. Oetjens, T. Kogel, and O. Bringmann, “Using SysML for modeling and generation of virtual platforms,” presented at the SNUG Europe 2019 Proceedings, 2019, p. 26.
- [37] A. Kirchner, J.-H. Oetjens, and O. Bringmann, “Using SysML for modelling and code generation for smart sensor ASICs,” in *2018 Forum on Specification & Design Languages (FDL)*, Garching: IEEE, Sep. 2018, pp. 5–16, ISBN: 978-1-5386-6418-6. DOI: 10.1109/FDL.2018.8524051. [Online]. Available: <https://ieeexplore.ieee.org/document/8524051/> (visited on 08/23/2019).
- [38] N. Prakash and D. Prakash, *Data Warehouse Requirements Engineering a decision based approach*. Singapore: Springer Singapore, 2018, ISBN: 978-981-10-7018-1 978-981-10-7019-8. DOI: 10.1007/978-981-10-7019-8. [Online]. Available: <http://link.springer.com/10.1007/978-981-10-7019-8> (visited on 02/19/2020).
- [39] R. R. Young, *The Requirements Engineering Handbook*. Artech House, 2004, 288 pp., Google-Books-ID: Rkulp4N3JsC, ISBN: 978-1-58053-618-9.
- [40] E. O. for Civil Aviation Equipment. (). EUROCAE-european organization for civil aviation equipment, [Online]. Available: <https://global.ihs.com/standards.cfm?publisher=EUROCAE> (visited on 02/19/2020).
- [41] Radio Technical Commission for Aeronautics. (). Home | RTCA, [Online]. Available: <https://www.rtca.org/> (visited on 02/19/2020).
- [42] ISO/TC 22/SC 32 Electrical and electronic components and general system aspects and Electrical and electronic components and general system aspects, *ISO 26262-1:2018(en), Road vehicles — Functional safety — Part 1: Vocabulary*, Dec. 2018. [Online]. Available: <https://www.iso.org/obp/ui/#iso:std:iso:26262:-1:ed-2:v1:en> (visited on 03/01/2020).
- [43] P. David and M. Shawky, “Supporting ISO 26262 with SysML, benefits and limits,” ESREL 2010, p. 9, Sep. 2010. [Online]. Available: <https://hal.archives-ouvertes.fr/hal-00579540> (visited on 01/19/2020).

- [44] A. Guegan and A. Bonnaud, “Assessing the maturity of interface design,” in *Complex systems design & management: Proceedings of the Ninth International Conference on Complex Systems Design & Management CSD&M Paris 2018*, E. Bonjour, D. Krob, L. Palladino, and F. Stephan, Eds., New York, NY: Springer Nature Switzerland AG, 2019, pp. 56–66, ISBN: 978-3-030-04208-0. DOI: 10.1007/978-3-030-04209-7\_5.
- [45] L. Karlsson, s. G. Dahlstedt, B. Regnell, J. Natt och Dag, and A. Persson, “Requirements engineering challenges in market-driven software development – an interview study with practitioners,” *Information and Software Technology*, vol. 49, no. 6, pp. 588–604, Jun. 2007, ISSN: 09505849. DOI: 10.1016/j.infsof.2007.02.008. [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/S0950584907000183> (visited on 03/04/2020).
- [46] F. Stallinger, R. Neumann, R. Schossleitner, and R. Zeilinger, “Linking software life cycle activities with product strategy and economics: Extending iso/iec 12207 with product management best practices,” in *Software Process Improvement and Capability Determination: 11th International Conference, SPICE 2011, Dublin, Ireland, May 30 - June 1, 2011. Proceedings*, R. V. O'Connor, T. Rout, F. McCaffery, and A. Dorling, Eds., Springer Verlag, 2011, pp. 157–168, ISBN: 978-3-642-21232-1.
- [47] A. R. Silva and M. Rosemann, “Design principles for inter-organizational systems development – case hanel,” in *Design science research in information systems: advances in theory and practice: 7th international conference, DESRIST 2012, Las Vegas, NV, USA, May 14-15, 2012: proceedings*, K. Pefers, M. Rothenberger, and W. Kuechler, Eds., ser. Lecture notes in computer science, OCLC: ocn785082591, vol. 7286, Berlin, Heidelberg: Springer, 2012, pp. 271–286, ISBN: 978-3-642-29862-2. DOI: 10.1007/978-3-642-29863-9. [Online]. Available: <http://link.springer.com/10.1007/978-3-642-29863-9>.
- [48] S. A. Fricker, C. Thümmler, and A. Gavras, Eds., *Requirements Engineering for Digital Health*, Cham, Switzerland: Springer International Publishing, 2015, ISBN: 978-3-319-09797-8 978-3-319-09798-5. DOI: 10.1007/978-3-319-09798-5. [Online]. Available: <http://link.springer.com/10.1007/978-3-319-09798-5> (visited on 02/06/2020).
- [49] R. J. Wieringa and J. M. G. Heerkens, “The methodological soundness of requirements engineering papers: A conceptual framework and two case studies,” *Requirements Engineering*, vol. 11, no. 4, pp. 295–307, Sep. 2006, ISSN: 0947-3602, 1432-010X. DOI: 10.1007/s00766-006-0037-6. [Online]. Available: <http://link.springer.com/10.1007/s00766-006-0037-6> (visited on 02/22/2020).
- [50] C. Haskins, Ed., *SYSTEMS ENGINEERING HANDBOOK : A GUIDE FOR SYSTEM LIFE CYCLE PROCESSES AND ACTIVITIES*, INCOSE-TP-

- 2003-002-03, Jun. 2006. [Online]. Available: <https://www.incose.org/systems-engineering> (visited on 02/26/2020).
- [51] L. Wen, D. Tuffley, and T. Rout, "Using composition trees to model and compare software process," in *Software Process Improvement and Capability Determination: 11th International Conference, SPICE 2011, Dublin, Ireland, May 30 - June 1, 2011. Proceedings*, R. V. O'Connor, T. Rout, F. McCaffery, and A. Dorling, Eds., Springer Verlag, 2011, pp. 1–15, ISBN: 978-3-642-21232-1.
- [52] A. Drechsler, "Design science as design of social systems – implications for information systems research," in *Design science research in information systems: advances in theory and practice: 7th international conference, DESRIST 2012, Las Vegas, NV, USA, May 14-15, 2012: proceedings*, K. Peffers, M. Rothenberger, and W. Kuechler, Eds., ser. Lecture notes in computer science, OCLC: ocn785082591, vol. 7286, Berlin, Heidelberg: Springer, 2012, pp. 191–205, ISBN: 978-3-642-29862-2. DOI: 10.1007/978-3-642-29863-9. [Online]. Available: <http://link.springer.com/10.1007/978-3-642-29863-9>.
- [53] L. Delligatti, *SysML Distilled: A Brief Guide to the Systems Modeling Language*. Addison-Wesley, Nov. 8, 2013, 304 pp., Google-Books-ID: 3bMJA-gAAQBAJ, ISBN: 978-0-13-343033-2.
- [54] OMG<sup>®</sup>, *OMG<sup>®</sup> Unified Modeling Language version 2.5*, Mar. 1, 2015. [Online]. Available: <https://www.omg.org/spec/UML/2.5/PDF> (visited on 02/29/2020).
- [55] *OMG Systems Modeling Language (OMG<sup>®</sup> SysML<sup>®</sup>)*, Version 1.6, Nov. 2019. [Online]. Available: <https://www.omg.org/spec/SysML/1.6/PDF> (visited on 03/06/2020).
- [56] B. W. Boehm, *Software Engineering Economics*. Prentice-Hall, 1981, 800 pp., ISBN: 978-0-13-822122-5. Google Books: VphQAAAAMAAJ.
- [57] A. Kirchner, "Methodenübersicht", Fixed-term.Chukwuma.Onuoha@de.bosch.com, Apr. 17, 2020.
- [58] D. Reimann, *Projektplanung*, Aug. 2, 2006. [Online]. Available: <https://www.flickr.com/photos/dbloete/204793785/> (visited on 02/07/2020).
- [59] L. E. Hart, "Introduction to model-based system engineering (MBSE) and SysML," Presentation, INCOSE Chapter Meeting, Delaware Valley, Jul. 30, 2015, [Online]. Available: <https://www.incose.org/docs/default-source/delaware-valley/mbse-overview-incose-30-july-2015.pdf> (visited on 02/01/2020).
- [60] PivotPoint Technology Corp. (). MBSE works<sup>™</sup>: MBSE + SysML Overview - What is MBSE? MBSEworks.com. Library Catalog: mbseworks.com, [Online]. Available: <https://mbseworks.com/mbse-overview/index.html> (visited on 03/06/2020).

- [61] Object Management Group. (Oct. 21, 2019). MBSE wiki, [Online]. Available: <https://web.archive.org/web/20191021041347/http://www.omgwiki.org/MBSE/doku.php> (visited on 02/26/2020).
- [62] S. Friedenthal, A. Moore, and R. Steiner, “OMG systems modeling language (OMG SysML™) tutorial,” p. 132, Sep. 2009. [Online]. Available: <http://www.omgsysml.org/INCOSE-OMGSysML-Tutorial-Final-090901.pdf> (visited on 02/01/2020).
- [63] E. Barnhart. (Dec. 4, 2018). What is MBSE? | systems engineering, evolved, [Online]. Available: <https://web.archive.org/web/20181204101354/http://vmcse.com/2016/03/13/what-is-mbse-deriving-a-definition/> (visited on 02/26/2020).
- [64] C. Ebert, “Improving engineering efficiency with PLM/ALM,” *Softw Syst Model*, vol. 12, no. 3, pp. 443–449, Jul. 2013, ISSN: 1619-1366, 1619-1374. DOI: 10.1007/s10270-013-0347-3. [Online]. Available: <http://link.springer.com/10.1007/s10270-013-0347-3> (visited on 02/19/2020).
- [65] O. Maksimenkova and A. Neznanov, “Blended learning in software engineering education: The application lifecycle management experience with computer-supported collaborative learning,” in *2015 International Conference on Interactive Collaborative Learning (ICL)*, Florence: IEEE, Sep. 2015, pp. 655–662, ISBN: 9781479987078. DOI: 10.1109/ICL.2015.7318104. [Online]. Available: <http://ieeexplore.ieee.org/document/7318104/> (visited on 02/27/2020).
- [66] F. Usmani, *Configuration management vs change management*, Dec. 11, 2019. [Online]. Available: <https://pmstudycircle.com/2012/01/configuration-management-vs-change-management/> (visited on 02/22/2020).
- [67] A. Bucaioni, L. Addazi, A. Cicchetti, F. Ciccozzi, R. Eramo, S. Mubeen, and M. Sjödin, “MoVES: A model-driven methodology for vehicular embedded systems,” *IEEE Access*, vol. 6, pp. 6424–6445, 2018, Conference Name: IEEE Access, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2018.2789400.
- [68] OASIS Open Project. (Jan. 8, 2020). About | OSLC. Library Catalog: open-services.net, [Online]. Available: <https://web.archive.org/web/20200108164210/https://open-services.net/about/> (visited on 02/26/2020).
- [69] M. Elaasar and A. Neal, “Integrating modeling tools in the development lifecycle with OSLC: A case study,” in *16th International Conference, MODELS 2013 Model-Driven Engineering Languages and Systems*, A. Moreira, B. Schätz, J. Gray, A. Vallecillo, and P. Clarke, Eds., red. by D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos, D.

- Tygar, M. Y. Vardi, and G. Weikum, ser. Lecture Notes in Computer Science, vol. 8107, Miami, FL, USA: Springer Berlin Heidelberg, 2013, pp. 154–169, ISBN: 978-3-642-41532-6 978-3-642-41533-3. DOI: 10.1007/978-3-642-41533-3. [Online]. Available: <http://link.springer.com/10.1007/978-3-642-41533-3> (visited on 02/22/2020).
- [70] D. Emery. (Apr. 7, 2019). Standards, APIs, interfaces and bindings, [Online]. Available: <https://web.archive.org/web/20190407090846/http://oldwww.acm.org/tsc/apis.html> (visited on 02/26/2020).
- [71] How to Do in Java (Blogs). (Feb. 21, 2020). What is REST – learn to create timeless REST APIs. Library Catalog: restfulapi.net, [Online]. Available: <https://restfulapi.net/> (visited on 02/26/2020).
- [72] R. T. Fielding, “In information and computer science,” PhD thesis, 2000. [Online]. Available: [https://www.ics.uci.edu/~fielding/pubs/dissertation/fielding\\_dissertation.pdf](https://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf) (visited on 02/26/2020).
- [73] Mulesoft. (Aug. 22, 2019). What is REST API design? | MuleSoft, MuleSoft, [Online]. Available: <https://web.archive.org/web/20190822063738/https://www.mulesoft.com/resources/api/what-is-rest-api-design> (visited on 02/26/2020).
- [74] M. I. Kamata and T. Tamai, “How does requirements quality relate to project success or failure?” In *15th IEEE International Requirements Engineering Conference (RE 2007)*, ISSN: 2332-6441, Oct. 2007, pp. 69–78. DOI: 10.1109/RE.2007.31.
- [75] PwC, “Opportunities for the global semiconductor market Growing market share by embracing AI,” Tech. Rep., 2019.
- [76] O. Gotel and C. Finkelstein, “An analysis of the requirements traceability problem,” in *Proceedings of IEEE International Conference on Requirements Engineering*, Colorado Springs, CO, USA: IEEE Comput. Soc. Press, 1994, pp. 94–101, ISBN: 978-0-8186-5480-0. DOI: 10.1109/ICRE.1994.292398. [Online]. Available: <http://ieeexplore.ieee.org/document/292398/> (visited on 02/22/2020).
- [77] J. Dick, E. Hull, and K. Jackson, *Requirements Engineering*. Cham, Switzerland: Springer International Publishing, 2017, ISBN: 978-3-319-61072-6 978-3-319-61073-3. DOI: 10.1007/978-3-319-61073-3. [Online]. Available: <http://link.springer.com/10.1007/978-3-319-61073-3> (visited on 02/06/2020).
- [78] IBM Knowledge Center, Pittsburgh. (Oct. 24, 2014). Rational solution for collaborative lifecycle management v6.0.6.1 documentation. Library Catalog: www.ibm.com, [Online]. Available: [https://www.ibm.com/support/knowledgecenter/SSYMRC\\_6.0.6.1/com.ibm.rational.clm.doc/helpindex\\_clm.html](https://www.ibm.com/support/knowledgecenter/SSYMRC_6.0.6.1/com.ibm.rational.clm.doc/helpindex_clm.html) (visited on 02/05/2020).

- [79] IBM Knowledge Center, Pittsburgh. (May 1, 2007). IBM Developer : New to Rational, [Online]. Available: <https://web.archive.org/web/20191106064741/https://www.ibm.com/developerworks/rational/newto/index.html> (visited on 02/16/2020).
- [80] IBM Knowledge Center, Pittsburgh. (Oct. 24, 2014). Overview of rational DOORS next generation, [Online]. Available: [https://www.ibm.com/support/knowledgecenter/SSYQBZ\\_9.6.1/com.ibm.doors.requirements.doc/topics/c\\_overview\\_dng.html](https://www.ibm.com/support/knowledgecenter/SSYQBZ_9.6.1/com.ibm.doors.requirements.doc/topics/c_overview_dng.html) (visited on 02/19/2020).
- [81] IBM Knowledge Center, Pittsburgh. (Oct. 24, 2014). Overview of rational rhapsody, [Online]. Available: [https://www.ibm.com/support/knowledgecenter/SSB2MU\\_8.4.0/com.ibm.rhp.overview.doc/topics/rhp\\_c\\_po\\_rr\\_product\\_overview.html](https://www.ibm.com/support/knowledgecenter/SSB2MU_8.4.0/com.ibm.rhp.overview.doc/topics/rhp_c_po_rr_product_overview.html) (visited on 02/19/2020).
- [82] IBM Knowledge Center, Pittsburgh. (Oct. 24, 2014). Overview of rational team concert, [Online]. Available: [https://www.ibm.com/support/knowledgecenter/SSYMRC\\_6.0.6/com.ibm.team.concert.doc/topics/c\\_product-overview.html?pos=2](https://www.ibm.com/support/knowledgecenter/SSYMRC_6.0.6/com.ibm.team.concert.doc/topics/c_product-overview.html?pos=2) (visited on 02/19/2020).
- [83] IBM Knowledge Center, Pittsburgh. (Oct. 24, 2014). Rhapsody Model Manager. Library Catalog: [www.ibm.com](http://www.ibm.com), [Online]. Available: [https://www.ibm.com/support/knowledgecenter/SSYMRC\\_6.0.6/com.ibm.rational.rmm.overview.doc/com.ibm.rational.rmm.overview.doc\\_eclipse-gentopic1.html](https://www.ibm.com/support/knowledgecenter/SSYMRC_6.0.6/com.ibm.rational.rmm.overview.doc/com.ibm.rational.rmm.overview.doc_eclipse-gentopic1.html) (visited on 03/01/2020).
- [84] IBM Knowledge Center, Pittsburgh. (Oct. 24, 2014). Managing configurations of models in AM applications. Library Catalog: [www.ibm.com](http://www.ibm.com), [Online]. Available: [https://www.ibm.com/support/knowledgecenter/SSYMRC\\_6.0.6.1/com.ibm.jazz.vvc.doc/topics/t\\_mng\\_cfg\\_rmm\\_stub.html](https://www.ibm.com/support/knowledgecenter/SSYMRC_6.0.6.1/com.ibm.jazz.vvc.doc/topics/t_mng_cfg_rmm_stub.html) (visited on 03/01/2020).
- [85] IBM Knowledge Center, Pittsburgh. (Oct. 24, 2014). Overview. Library Catalog: [www.ibm.com](http://www.ibm.com), [Online]. Available: [https://www.ibm.com/support/knowledgecenter/SSYMRC\\_6.0.6/com.ibm.rational.rmm.overview.doc/topics/rhp\\_c\\_rmm\\_overview.html?pos=2](https://www.ibm.com/support/knowledgecenter/SSYMRC_6.0.6/com.ibm.rational.rmm.overview.doc/topics/rhp_c_rmm_overview.html?pos=2) (visited on 03/01/2020).
- [86] IBM Knowledge Center, Pittsburgh. (Oct. 24, 2014). Overview of jazz team server, [Online]. Available: [https://www.ibm.com/support/knowledgecenter/SSYMRC\\_6.0.6/com.ibm.help.common.jazz.calm.doc/topics/c\\_jts\\_overview.html](https://www.ibm.com/support/knowledgecenter/SSYMRC_6.0.6/com.ibm.help.common.jazz.calm.doc/topics/c_jts_overview.html) (visited on 02/19/2020).
- [87] IBM Knowledge Center, Pittsburgh. (Oct. 24, 2014). Project area, [Online]. Available: [https://www.ibm.com/support/knowledgecenter/SSYMRC\\_6.0.6.1/com.ibm.jazz.platform.doc/topics/c\\_project\\_area.html](https://www.ibm.com/support/knowledgecenter/SSYMRC_6.0.6.1/com.ibm.jazz.platform.doc/topics/c_project_area.html) (visited on 02/19/2020).

- [88] H.-P. Hoffmann, *Systems Engineering Best Practices with the Rational Solution for Systems and Software Engineering : Model-Based Systems Engineering with Rational Rhapsody and Rational Harmony for Systems Engineering*, Deskbook Release 4.1. IBM Corporation Software Group, Route 100 Somers, NY, U.S.A., 2014, 162 pp. [Online]. Available: [https://www.ibm.com/developerworks/community/groups/service/html/communityview?communityUuid=dbc39547-3619-4c31-9535-0b583a4e6190#{#}fullpageWidgetId=W62078615f88f{\\_-}4809{\\_-}afad{\\_-}c27cdc9d7e711{&}file=2132d88d-4dde-40b4-8102-254ca4456c82](https://www.ibm.com/developerworks/community/groups/service/html/communityview?communityUuid=dbc39547-3619-4c31-9535-0b583a4e6190#{#}fullpageWidgetId=W62078615f88f{_-}4809{_-}afad{_-}c27cdc9d7e711{&}file=2132d88d-4dde-40b4-8102-254ca4456c82).
- [89] IBM Knowledge Center, Pittsburgh. (Oct. 24, 2014). Context patterns. Library Catalog: [www.ibm.com](http://www.ibm.com/support/knowledgecenter/SSB2MU_8.4.0/com.ibm.rhp.matrix.doc/topics/r_context_patterns.html), [Online]. Available: [https://www.ibm.com/support/knowledgecenter/SSB2MU\\_8.4.0/com.ibm.rhp.matrix.doc/topics/r\\_context\\_patterns.html](https://www.ibm.com/support/knowledgecenter/SSB2MU_8.4.0/com.ibm.rhp.matrix.doc/topics/r_context_patterns.html) (visited on 03/22/2020).
- [90] R. V. O'Connor, T. Rout, F. McCaffery, and A. Dorling, Eds., *Software Process Improvement and Capability Determination: 11th International Conference, SPICE 2011, Dublin, Ireland, May 30 - June 1, 2011. Proceedings*, Springer Verlag, 2011, ISBN: 978-3-642-21232-1.
- [91] K. Peffers, M. Rothenberger, and W. Kuechler, Eds., *Design science research in information systems: advances in theory and practice: 7th international conference, DESRIST 2012, Las Vegas, NV, USA, May 14-15, 2012: proceedings*, vol. 7286, Lecture notes in computer science 7286, OCLC: ocn785082591, Berlin, Heidelberg: Springer, 2012, 438 pp., ISBN: 978-3-642-29862-2. DOI: 10.1007/978-3-642-29863-9. [Online]. Available: <http://link.springer.com/10.1007/978-3-642-29863-9>.



TECHNISCHE UNIVERSITÄT  
CHEMNITZ

Studentenservice – Zentrales Prüfungsamt  
Selbstständigkeitserklärung

Name: Onuoha Vorname: Chukwuma Onuoha geb. am: 15.04.1983 Matr.-Nr.: 458304	<b>Bitte beachten:</b> 1. Bitte binden Sie dieses Blatt am Ende Ihrer Arbeit ein.
--	--

Selbstständigkeitserklärung\*

Ich erkläre gegenüber der Technischen Universität Chemnitz, dass ich die vorliegende **Masterarbeit** selbstständig und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel angefertigt habe.

Die vorliegende Arbeit ist frei von Plagiaten. Alle Ausführungen, die wörtlich oder inhaltlich aus anderen Schriften entnommen sind, habe ich als solche kenntlich gemacht.

Diese Arbeit wurde in gleicher oder ähnlicher Form noch nicht als Prüfungsleistung eingereicht und ist auch noch nicht veröffentlicht.

Datum: 30.07.2020

Unterschrift:  .....

\* Statement of Authorship

I hereby certify to the Technische Universität Chemnitz that this thesis is all my own work and uses no external material other than that acknowledged in the text.

This work contains no plagiarism and all sentences or passages directly quoted from other people's work or including content derived from such work have been specifically credited to the authors and sources.

This paper has neither been submitted in the same or a similar form to any other examiner nor for the award of any other degree, nor has it previously been published.