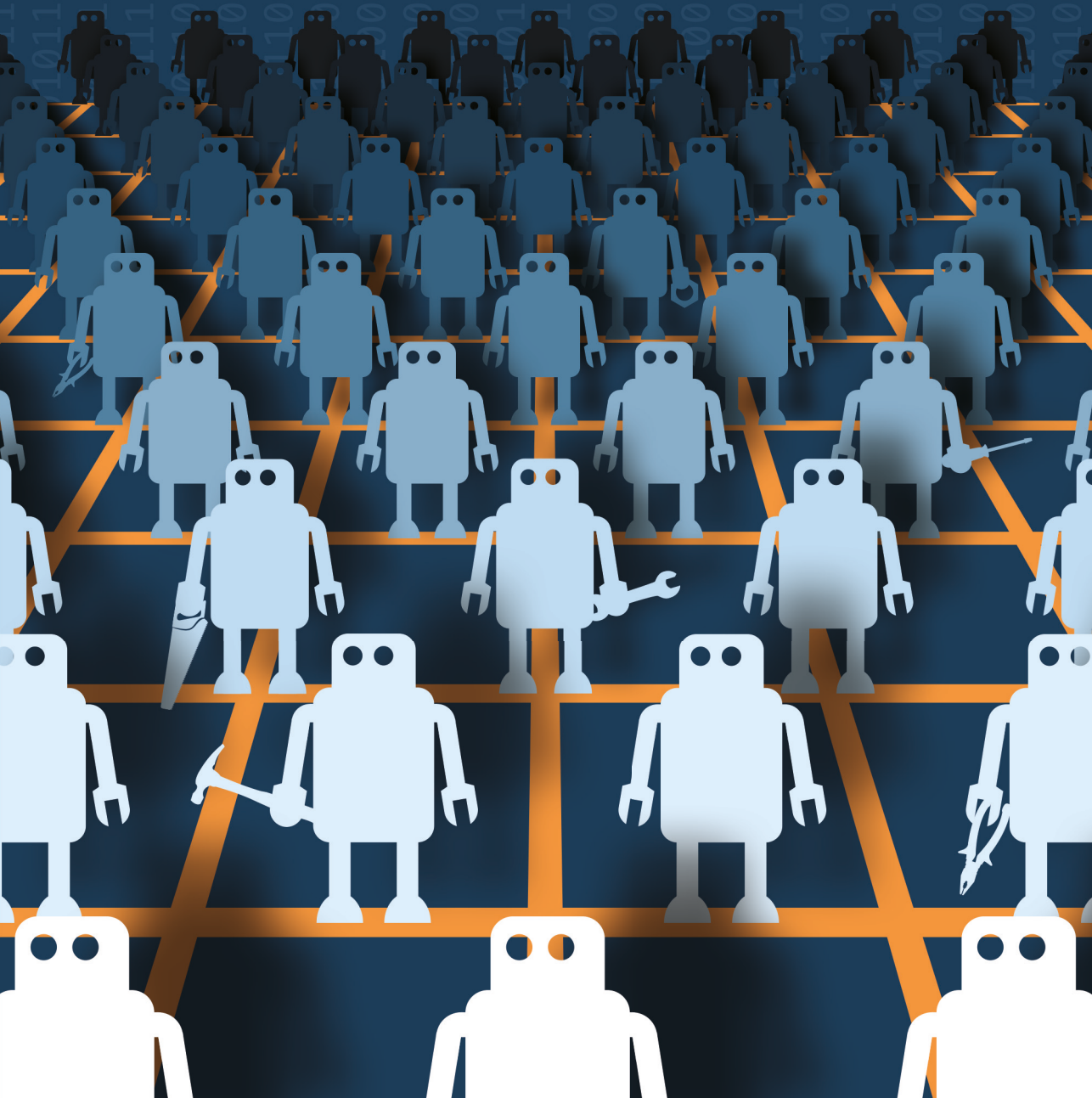
GRID MANUFACTURING

A CYBER-PHYSICAL APPROACH WITH AUTONOMOUS PRODUCTS AND RECONFIGURABLE MANUFACTURING MACHINES

DANIËL TELGEN



Grid Manufacturing

A Cyber-Physical Approach with Autonomous Products and Reconfigurable Manufacturing Machines

Daniël Telgen

SIKS Dissertation Series No. 2017-03. The research reported in this thesis has been carried out under the auspices of SIKS, the Dutch Research School for Information and Knowledge Systems.

The research in this thesis was performed at the Intelligent Systems group of the Utrecht University, the Netherlands.

The creation of this Thesis was financially supported by the HU University of Applied Sciences Utrecht, the Netherlands.

Final Version as of 15 december 2016

ISBN: 978-94-6299-517-8

Cover design: Design Your Thesis | www.designyourthesis.com
Printing: Ridderprint B.V. | www.ridderprint.nl
General Layout: Daniël Telgen

GRID MANUFACTURING

A CYBER-PHYSICAL APPROACH WITH AUTONOMOUS PRODUCTS AND RECONFIGURABLE MANUFACTURING MACHINES

Grid Productiesystemen
Een Cyber-Fysieke Aanpak met Autonome Producten en Herconfigureerbare Productiesystemen
(met een samenvatting in het Nederlands)

PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Universiteit Utrecht op gezag van de
rector magnificus, prof.dr. G.J. van der Zwaan,
ingevolge het besluit van het college voor promoties
in het openbaar te verdedigen op
dinsdag 31 januari 2017 des middags te 4.15 uur

door

Daniël Harold Telgen

geboren op 8 november 1981
te Rotterdam

Promotor: Prof. dr. J-J.Ch. Meyer

Abbreviations and Acronyms

3D	3-Dimensional - in the context of location: X,Y,Z position
6D	6-Dimensional - In context of location: X,Y,Z, Roll, Pitch, Yaw
AM	Agile Manufacturing
AMS	Agent Management Service
ATO	Assemble To Order
BDI	Belief - Desire - Intention (Agent)
CA	Customer Attributes (Axiomatic Design)
CAD	Computer-Aided Design
CNC	Computer Numerical Control
COPD	Customer Order Decoupling Point
COTS	Component(s) Off The Shelf
CPS	Cyber-Physical Systems
CT	Container Table
DDS	Dedicated Manufacturing Systems
DF	Directory Facilitator
DOF	Degrees Of Freedom
DP	Design Parameters (Axiomatic Design)
EA	Equiplet Agent
ETO	Make-To-Order
EUPASS	Evolvable Ultra-Precision Assembly Systems
FIPA	Foundation for Intelligent Physical Agent
FMS	Flexible Manufacturing System
FR	Functional Requirements
GADT	Global Agent Descriptor table
GEM	Grid Equiplet Module (Hierarchy)
GM	Grid Manufacturing
HAL	Hardware Abstraction Layer
HU	Hogeschool Utrecht (HU University of Applied Sciences Utrecht)
HMS	Holonic Manufacturing System
IO	Input Output
IOT	Internet Of Things
LCMD	Line Cell Module Device (Hierarchy)
JADE	Java Development Environment
JSON	Javascript Object Notation
LADT	Local Agent Descriptor Table
MAAS	Manufacturing As A Service
MAS	Multi-Agent System
MEMS	Microelectromechanical Systems

MST	Micro Systems Technology
MTO	Make-To-Order
MTS	Make-To-Stock
PA	Product Agent
PCB	Printed Circuit Board
PV	Process Variables (Axiomatic Design)
R5-COP	Reconfigurable ROS-based Resilient Reasoning Robotic Cooperating Systems
RP	Research Design Parameter
REXOS	Reconfigurable EQuipletS Operating Systems (QS is changed to an X)
ROS	Robot Operating System
RMS	Reconfigurable Manufacturing System
RSQ	Research Sub Question
RQ	Research Question
SDF	Simulation Description Format
UU	Utrecht University
UX	User Experience

Contents

Abbreviations and Acronyms **v**

1 Introduction **1**
1.1 Research Motivation . . . 3
1.1.1 Technology Enablers . . . 5
1.2 General Research Overview . . . 7
1.2.1 Manufacturing Paradigms and Literature . . . 7
1.2.2 Key Interests . . . 11
1.3 Thesis Overview . . . 11

2 Research Design **17**
2.1 Research Approach . . . 17
2.2 Six Manufacturing Principles . . . 18
2.3 Problem Statement . . . 19
2.4 Research Design Parameters . . . 21
2.5 Research Questions . . . 23
2.6 Research Methodology . . . 23
2.7 Research Objectives . . . 24
2.8 Scope . . . 24
2.9 Structure of the Thesis . . . 25

3 The Concept of Grid Manufacturing **31**
3.1 Conceptual Idea of Grid Manufacturing . . . 31
3.2 Grid Manufacturing . . . 32
3.3 Requirements . . . 35
3.3.1 Motivation . . . 36
3.3.2 Customer Attributes . . . 38
3.3.3 Functional Requirements . . . 38
3.4 Fundamental Technologies . . . 41
3.4.1 Agent Technology in Manufacturing . . . 42
3.4.2 Cyber-Physical Systems . . . 44
3.5 Conclusion . . . 45

4 Object Awareness **49**
4.1 Simplifying Object Recognition . . . 49
4.1.1 Problem Description . . . 49
4.1.2 Requirements . . . 50
4.1.3 Additional Research Questions . . . 51

4.1.4 Research Design Parameters . . . 51
4.1.5 Research Objectives . . . 51
4.1.6 Literature Overview . . . 52
4.2 Metadata in the Grid . . . 53
4.3 Proposal - Generic Method . . . 55
4.4 Design . . . 56
4.4.1 Requirement decomposition . . . 56
4.5 Implementation . . . 57
4.5.1 Preprocess . . . 59
4.5.2 Modelling Tool . . . 59
4.5.3 Vision Module . . . 61
4.5.4 6D Process . . . 64
4.6 Proof of concept: Floe . . . 65
4.6.1 Accuracy . . . 68
4.6.2 Limitations . . . 68
4.7 Discussion . . . 68
4.8 Conclusion . . . 69

5 Reconfiguration 75
5.1 Problem Description . . . 76
5.2 Additional Research Questions . . . 76
5.3 Literature Overview . . . 77
5.4 The Concept of Product Step Translations . . . 78
5.4.1 Capabilities . . . 80
5.5 Simplified overview of the Manufacturing Process . . . 81
5.5.1 Capability Checking . . . 82
5.5.2 Step Translations . . . 82
5.5.3 Scheduling . . . 83
5.5.4 Step Execution . . . 84
5.5.5 Product Steps . . . 85
5.5.6 Composite Steps . . . 86
5.5.7 Hardware Steps . . . 87
5.6 Offering Services . . . 89
5.7 Module Configuration . . . 89
5.8 Software Design . . . 91
5.8.1 HAL Overview . . . 92
5.8.2 Translation and Execution Process . . . 93
5.9 Implementation . . . 95
5.9.1 Knowledge Database . . . 95
5.9.2 HAL Implementation . . . 98
5.10 Discussion . . . 99
5.11 Future Work . . . 101

5.12 Conclusion . . . 101

6 Architecture **107**
6.1 Problem Description . . . 107
6.2 Research Question . . . 107
6.3 Design . . . 108
6.4 Choice of Technology . . . 109
6.5 Choice of Platforms . . . 110
6.5.1 Agent Platforms . . . 111
6.5.2 Diverse Platforms . . . 112
6.6 Reconfigurable Equiplets Operating System . . . 112
6.7 Implementation . . . 114
6.7.1 Middleware . . . 115
6.7.2 Grid . . . 115
6.7.3 Equiplets . . . 116
6.7.4 Modules . . . 118
6.7.5 Basic Operation . . . 123
6.8 Evaluation and Performance . . . 124
6.8.1 Synthetic testing . . . 125
6.8.2 Simulated Testing . . . 125
6.9 Conclusion . . . 128

7 System Behaviour **135**
7.1 Problem Description . . . 135
7.2 Additional Research Questions . . . 136
7.3 Software Control . . . 136
7.3.1 MAST . . . 137
7.3.2 Related Work . . . 137
7.3.3 MAST Problems . . . 137
7.3.4 Predictable Behaviour . . . 138
7.3.5 Proposal . . . 138
7.3.6 Hardware Representation . . . 139
7.3.7 Module States . . . 140
7.3.8 Modes . . . 141
7.3.9 States Throughout the Grid . . . 144
7.3.10 Discussion . . . 145
7.3.11 Machine State Conclusions . . . 146
7.4 Physical Safety . . . 147
7.4.1 Problem Description . . . 148
7.4.2 Proposal . . . 148
7.4.3 Sub Research Questions . . . 148
7.4.4 RQ4g - Safety Concerns . . . 149

7.4.5 Simulation Engine . . . 149
7.4.6 RQ4h - Adding New Models to the Simulation . . . 151
7.4.7 RQ4i - Integrating Safety Checks . . . 153
7.4.8 Implementation . . . 156
7.4.9 RQ4j - Simulation Integration . . . 157
7.4.10 Simulation Opportunities . . . 161
7.4.11 Physical Safety Conclusions . . . 162
7.4.12 Future Work . . . 164
7.5 Discussion . . . 165
7.6 Future Opportunities . . . 165
7.7 Conclusion . . . 166

8 Validation and Utilisation 171
8.1 Additional Research Questions . . . 171
8.2 Research Objectives . . . 172
8.3 Proactive Top-Down Usage of a Grid . . . 172
8.3.1 Simulation . . . 174
8.3.2 Top Down Management . . . 180
8.3.3 Discussion . . . 186
8.3.4 Conclusion for Proactive - Top-down Approach . . . 186
8.4 Reactive Aspects . . . 187
8.4.1 Problem Description . . . 187
8.4.2 Platform Description . . . 188
8.4.3 Hypothesis . . . 189
8.4.4 Mechanics . . . 189
8.4.5 Simulation Model . . . 190
8.4.6 Implementation . . . 194
8.4.7 Results . . . 194
8.4.8 Discussion . . . 198
8.4.9 Conclusion Reactive . . . 199
8.5 Future Work . . . 199
8.6 Conclusion . . . 200

9 Discussion 205
9.1 General Evaluation . . . 205
9.2 Review of the Research Design Parameters . . . 206

10 Conclusion 213
10.1 Review of the Research Questions . . . 213
10.2 Final Conclusion . . . 215

References 217

Abstract **227**

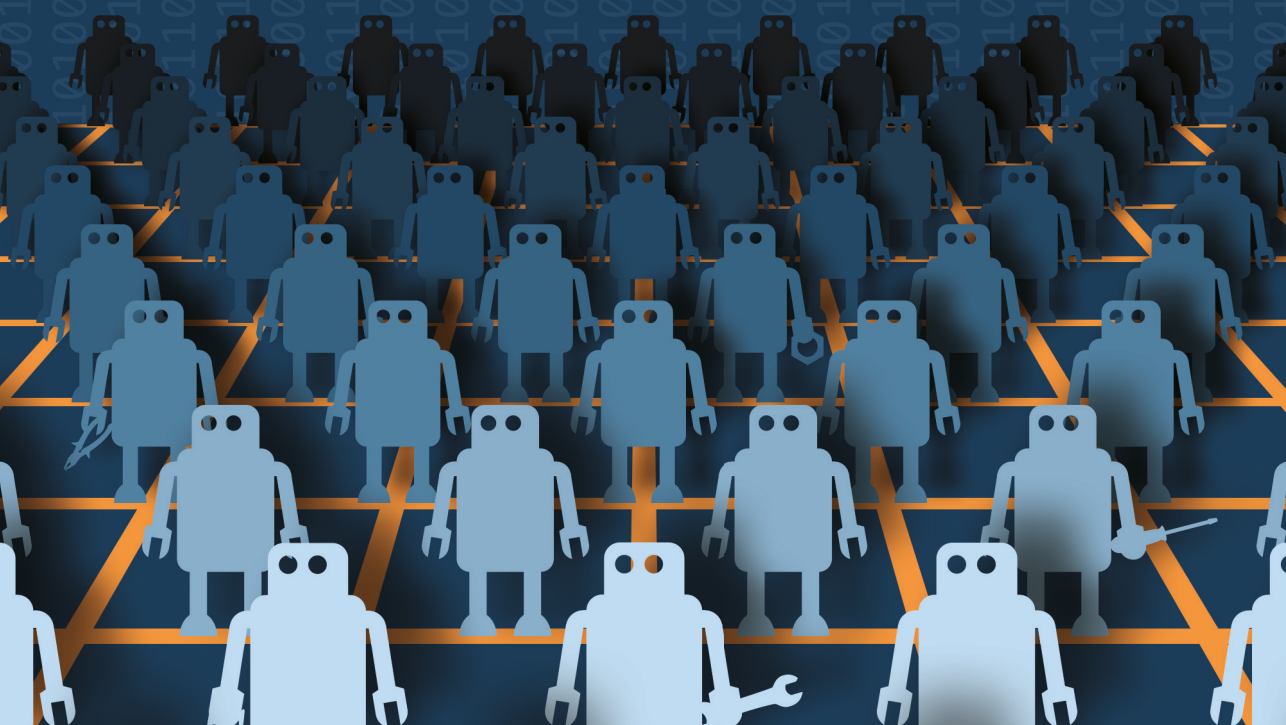
Dutch Summary **229**

Curriculum Vitae **231**

Acknowledgements **233**

SIKS Dissertation Series **237**

01



Introduction

“In the twenty-first century, the robot will take the place which slave labour occupied in ancient civilization.”

– Nikola Tesla

Introduction

Computers are increasingly changing industry. Whereas in the past this has created opportunities for improved logistics and overview like SCADA (Supervisory Control and Data Acquisition) it is now starting to change the industry itself (Telgen et al., 2013c). Computers are not only supporting existing mechatronical systems, but are fundamentally changing the processes in how they are used. This is the basis for many changes in industry and specifically the field of manufacturing. These technological changes and paradigms are known under a variety of names, including 'Smart Industry', 'Agile Manufacturing', 'Industry 4.0', and Cyber-Physical systems. Especially the ideas behind Cyber-Physical Systems and Agile Manufacturing were some of the original drivers behind this research. Whereas mechatronics was the integration of electronics, mechanical systems and computers into embedded devices, Cyber-Physical Systems can be seen as the next step: The integration of the internet and the virtual 'Cyber' world. With Cyber-Physical Systems, mechatronical devices become connected through the Internet of Things and use their virtual 'modelled' representation of the world to sense and act in the physical domain.

The opportunities that arise through new technologies like Cyber-Physical Systems give a new perspective on the principles and opportunities in view of Agile Manufacturing (AM). AM is designed to quickly respond to customer and market changes, while still controlling cost and quality (Gunasekaran, 1999).

The change to Cyber-Physical Systems comes with the use of more computing power. As in other areas, the increase in computing power also makes it easier in the manufacturing industry to integrate 'intelligent' behaviour. Dynamic (intelligent) behaviour can be made possible with the use of microsystems, i.e. sensors and actuators, together with advanced software to interpret the sensed data and act accordingly. In such a way a robotic system is created that can dynamically interact with its environment. However, the dynamic behaviour of such a system can also increase complexity (Telgen et al., 2012). Therefore, it is important to create a balanced architecture that, on the one hand, has a high performance to control the hardware, i.e. dynamically interpret and interact with its environment in real-time, and on the other hand, is not so complex that it will be difficult to use and shows unexpected, i.e. unsafe or unwanted, behaviour.

From a hardware perspective there are also a number of state-of-the-art developments. Systems become more dependent on multi-disciplinary fields;

integration of ICT, computer engineering, electronics and mechanical developments come at a cost and increase complexity. Since these costs have to be earned back, it is necessary to reuse systems as often as possible. Over the years the move to multi-disciplinary integration has already led to the creation of system engineering methods that focus on modular and autonomous systems. The modules can be used as building blocks that can be combined for a specific purpose. If these modular systems are standardised and well-documented, they can also be easily reconfigured to provide a range of options. If a well-defined module could be seen as an autonomous black box, it also lowers complexity of the overall system, where functionality can be abstracted on a higher level (Telgen et al., 2012).

All these developments have a high focus on flexibility in many forms: flexibility in manufacturing processes, in adapting the hardware and software, in the creation of the equipment itself and even in how products can be used. *Flexibility is the key* in the current developments. However, flexibility in manufacturing can be seen from many different perspectives. Hence, it is important to focus the research. *This PhD addresses the low-level technological means, i.e. not at the enterprise or resource level, but the control and device level to be able to automate the manufacturing of of products that at this time can commonly only be produced by hand.* To explain this further, please consider the classic automation pyramid, seen in Figure 1.1. From the perspective of the classic automation pyramid, the research will focus on the bottom three layers, i.e.: (1) the hardware level, with devices like sensors and actuators; (2) the module (low-level control) layer, with PCs or micro-controllers that connect to PLCs and PIDs to control the devices; (3) the Supervised Control and Data Acquisition Layer (SCADA) systems that connects and controls the systems from a higher perspective. The upper two levels, like the Manufacturing Execution System (MES), that for instance contains most manufacturing floor scheduling systems, and the Enterprise Resource Planning (ERP) are largely taken out of scope unless they are directly required or affected. For instance, scheduling is discussed to the extent (but no further) to show that it is required by the proof of concepts and affected by the possibilities that have been created throughout this research. However, the work of Moergestel (2014) does discuss scheduling and is compliant with the work performed in this research. More details on the scope and limitations will be discussed in the next chapter.

To be able to create flexibility at the technological level it is important to take an applied approach, to lower the complexity and create 'mature' technology that is transparent, i.e. understood, and can be adopted by industry.

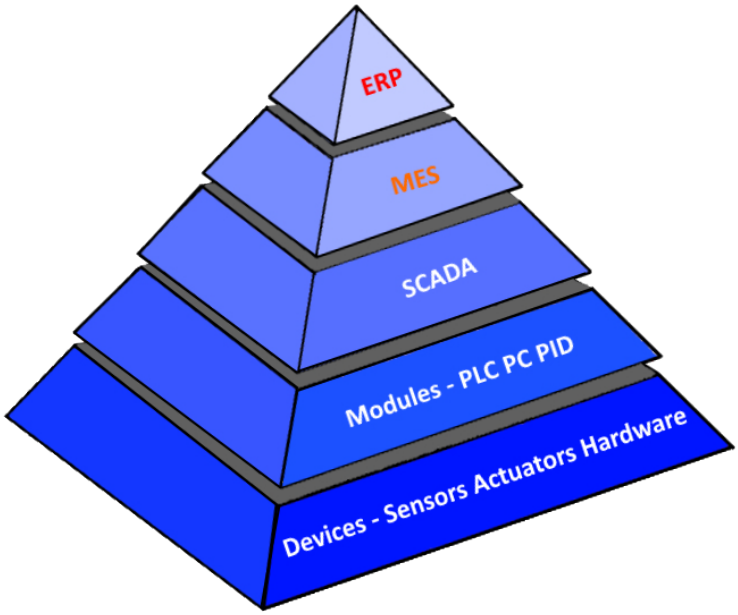


Figure 1.1: The classic automation pyramid.

1.1 Research Motivation

This research is motivated by the need for practicable, i.e. applicable, flexible systems that can dynamically change their purposes based on the current demand. The research will be focused on the technical aspects, based on the perspective of Cyber-Physical Systems, i.e. seen from the software but with a small overlap with the mechatronical systems. To motivate these choices let's further investigate some of the aspects of the current changes that influence the manufacturing industry:

Due to the advantages of using smart (or intelligent) systems for agile manufacturing use, many companies and research groups are experimenting and conducting Research and Development with several projects. However, the success in industry itself has so far been limited (Leitão, 2009). This is due to a number of reasons, including the complexity of such systems with the high initial investment costs, the expertise needed to create such systems, and the difficulties to create a business case for dynamic, or flexible, systems. Schild and Bussmann show this in a case where a successful self-organised manufacturing system was set up at Daimler-Chrysler. They state that while the system was successfully implemented it was discontinued because a technical advantage is not always a measurable economical advantage (Schild and Bussmann, 2007). When continuing this research in this field, therefore, it

is important to take this aspect into account. In this research, this will be done by considering several techniques and requirements that will lower the complexity and hardware costs, and increasing the flexibility to even higher levels.

Besides industry itself, there is also a change in retail products. Mass customisation is slowly beginning to become a standard. Wind even introduces the concept of 'customerisation'. This is a new business strategy that combines personal marketing strategies with mass customisation. Figure 1.2 shows that customerisation comes through a combination of standardisation, personalisation and mass customisation. (Wind and Rangaswamy, 2001) mention that successful customerisation requires the integration of multiple processes, including operations and R&D. He also states that increasing the digital content of everything the company does is one of two critical aspects that should be considered.

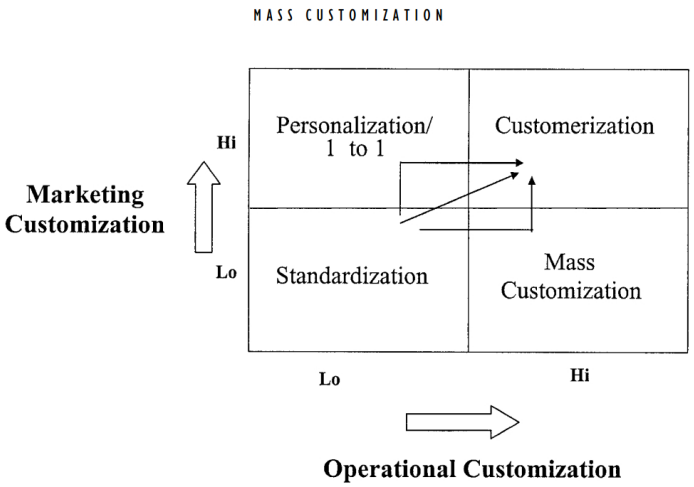


Figure 1.2: Combining personalisation and mass customisation towards a new business strategy (Wind and Rangaswamy, 2001).

Another aspect is the wish to automate high-mix, low-volume manufacturing. (Puik and Moergestel, 2010) already noted that the automated manufacturing of low volumes has been achieved in some fields, e.g., Printed Circuit Boards (PCB). However, the same has not been achieved for microsystems, which still largely rely on manual labour. Puik said: "If existing equipment would be gradually upgradable, in a true reconfigurable sense, investments in equipment could be reduced significantly".

ElMaraghy (2005) also shows the role of humans and automation in the evolution of manufacturing systems, this is shown in Figure 1.3. The current

research is focused strongly on the reconfiguration aspect. However, while not a purpose by itself, the Human/Machine systems interfacing focuses on a dynamic (changing) environment that cannot always be predicted. As such this is connected to the principle of 'flexibility'.

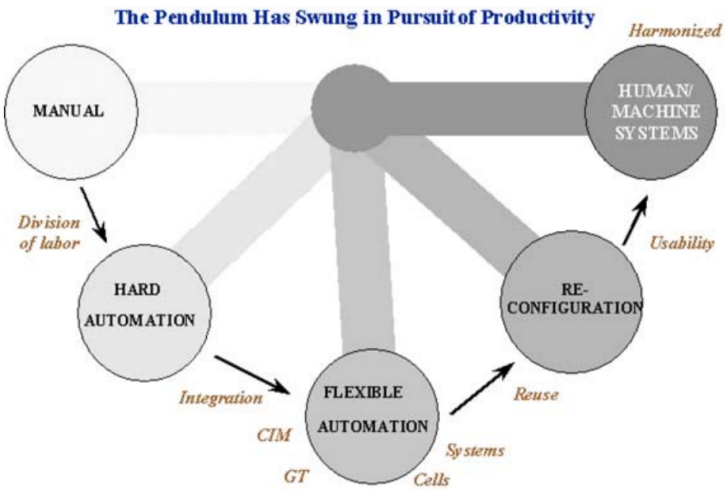


Figure 1.3: Role of humans and automation in the evolution of manufacturing systems (ElMaraghy, 2005).

From this preliminary work a number of technology enablers were identified that require some further introduction:

1.1.1 Technology Enablers

There are many technology enablers that influence the field of agile manufacturing. Therefore the introduction chapter shall first discuss five of these enablers.

Internet of Things

The internet of things is an important paradigm in this field. The idea of physical (autonomous) objects that are connected provides many opportunities. Kortuem even envisioned Smart Objects as building blocks for the internet of things (Kortuem et al., 2010). He defined *smart objects* as autonomous physical/digital objects with sensing, processing, and network capabilities. The idea of distributing intelligence between different entities could lower the complexity of the overall system and reduce the complexity by isolating specific functionalities.

Agent Technology

The word Agent comes from the Latin word *agere*, which means: to act. Software agents, as shown in Figure 1.4 are entities that have their own interpretation of their environment on which they act autonomously. Hence, we uphold the definition of Wooldridge and Jennings: 'An agent is an encapsulated computer system that is situated in some environment and that is capable of flexible, autonomous action in that environment in order to meet its design objectives' (Wooldridge and Jennings, 1995).

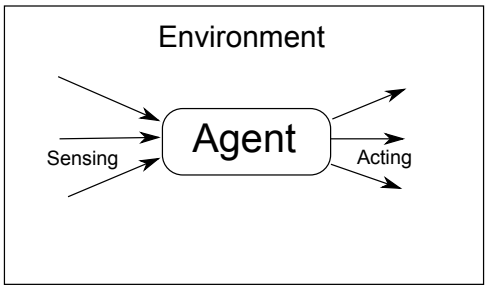


Figure 1.4: An autonomous agent in its environment.

Agents can be seen as 'objects with an attitude', since unlike an object an agent has control over its own behaviour. Agents can also be seen as a higher abstraction of objects, which makes them ideally suited to deal with complex dynamic environments.

Microsystems

Microsystems, Micro System Technology (MST) or Microelectromechanical Systems (MEMS), i.e. sensors and actuators, are not only interesting as a product for manufacturing. MST is one of the enablers for flexible systems. Akhras even speaks of smart systems as devices and materials that could mimic human muscular and nervous systems. Smart Systems would consist of systems with sensors and actuators that would be embedded or attached as an integral part of a system (Akhras, 1997). From this perspective MST is not only interesting as a specific product to be manufactured in high-mix, low-volume quantities, but also as an enabler for the flexible manufacturing systems themselves. The use of MST could be essential to deal with a changing environment to create manufacturing systems that are both flexible and safe.

Robotics

Robots are introduced in many forms and purposes; from being able to clean the floor until assembling a car. The word *robot* was introduced in a play in

1921 and is derived from the Czech word *robota*, which stands for servitude or forced labour. Robots were originally seen as autonomous machines that could substitute a human to some extent. Hence, Robotics is an interdisciplinary field that borders on multiple engineering principles, including mechanical, electrical, and computer engineering. Robotics are an important enabler in manufacturing, and are having an increasing impact on industry and society in general.

Additive Manufacturing

Additive Manufacturing, e.g., 3D printing, is seen as another enabler for flexible or 'agile' manufacturing. Like MST, Additive Manufacturing is an enabler for both customer products and the manufacturing equipment itself. The ability to almost instantly print 3D objects is ideal for use in prototypes and to adapt modules for different dimensions. Therefore Additive Manufacturing is a strong enabler for agile manufacturing that quickly needs to adapt to new needs.

1.2 General Research Overview

This section provides a theoretical framework related to the current research. It discusses a number of paradigms and technologies that will form the basis, where in the next chapter the specific problem and research description will be discussed.

1.2.1 Manufacturing Paradigms and Literature

Many paradigms have emerged that are of influence in the manufacturing industry, but the most influential have been the three main paradigms, shown in Figure 1.5.

Dedicated Manufacturing Systems (DMS) are the classic way of mass-production. In this paradigm, all manufacturing systems are developed for a specific single-purpose goal with limited to no dynamic properties. This creates a cost-efficient system for producing high volumes of a single product over a longer timespan (Koren and Shpitalni, 2010). The requirements stay the same, therefore, DMS are known to have a high performance and limited initial costs.

Flexible Manufacturing Systems (FMS) offer dynamic behaviour, which they use to react to changes. Usually, Flexible Manufacturing Systems offer a single purpose where the machine has the ability to perform one action. This ability is combined with sensors, like a vision or dynamic routing system, so

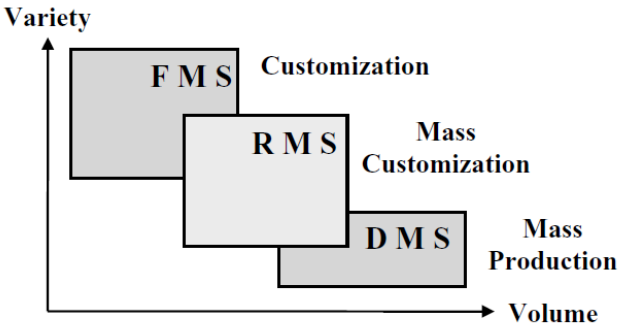


Figure 1.5: The three main manufacturing paradigms (HU, 2005).

it can adapt certain parameters, e.g. the position of a product. This creates more flexibility at the disadvantage of complexity and cost to the initial implementations.

Reconfigurable Manufacturing Systems (RMS) are unique in the perspective that the functionality of the system itself can be adapted (Mehrabi et al., 2000). As shown in Figure 1.5, they are positioned between FMS and DMS. However, since RMS provide a way to integrate change within a system it is expected that they will become more flexible over time (Stecke, 2005). ElMaraghy notes that the key characteristics of RMS include modularity, integrability, scalability, convertibility, and diagnosability (ElMaraghy, 2005). These characteristics have to be taken into account when defining the requirements for a flexible manufacturing architecture.

Besides the main three main manufacturing paradigms, there are also many other paradigms and methods of interest.

Agile Manufacturing (AM), is characterised by the integration of customer and supplier for both product design, as well as manufacturing, marketing, and support services (Gunasekaran, 1999). An agile manufacturing environment creates processes, tools, and a knowledge base to enable the organisation to respond quickly to the customer needs and market changes whilst still controlling costs and quality (Koh and Wang, 2010).

Manufacturing As a Service (MAAS) is a concept to deliver customisable and on-demand manufacturing. This is closely related to manufacturing clouds, where factories and their IT infrastructure are interconnected to create an infrastructure where ad-hoc products are being made (Rauschecker et al., 2014).

Holonic Manufacturing is seen as an alternative to hierarchical management of manufacturing systems. It focuses on modularisation and 'plug and play' capabilities when developing or using manufacturing systems (Mcfarlane

and Bussmann, 2000). Holonic manufacturing systems are often implemented using Multi-Agent Systems (Giret and Botti, 2009).

Noteworthy are also a number of concepts that are related to this research:

Smart Industry and **Industry 4.0** are often used as a concept that combines industrial systems with properties like the 'internet of things' and other cloud-related services. The name 'Industries 4.0' is based on the idea that this could be the 4th industrial revolution, see Figure 1.6

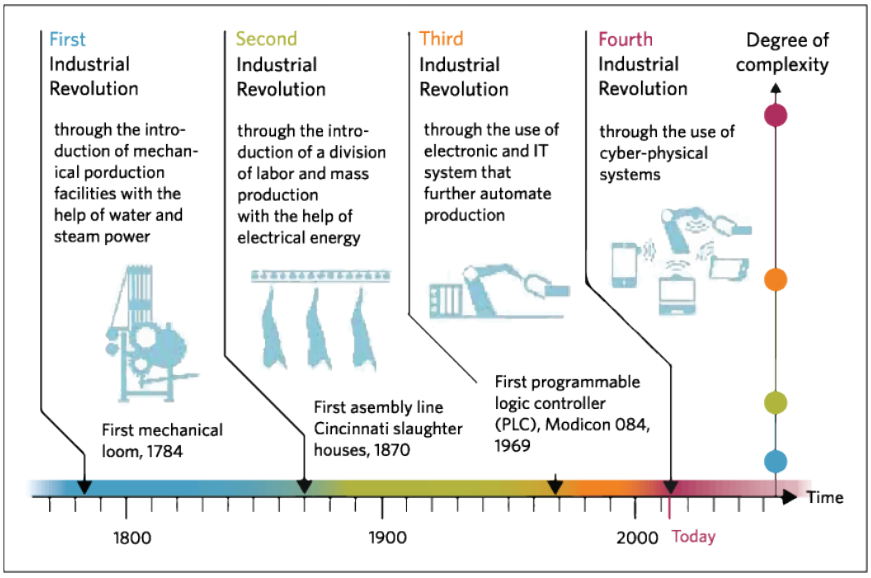


Figure 1.6: Industry 4.0 - source: DKFI - German Research Center for Artificial Intelligence (2011).

These depict the vision of smart factories that consist of **Cyber-Physical Systems**. As mentioned before, Cyber-Physical Systems are collaborating computational (virtual) elements that control physical entities. In other words, a virtual entity with its virtual world image that uses that image to control and interpret the physical world and operate a physical counterpart in this environment. This virtual entity is commonly an embedded system within the system that it controls. (Lee et al., 2015) extends this idea even further by proposing a new architecture for CPS based on 5 layers, see Figure 1.7.

The proposed layers also reflect on some of the other aspects mentioned in this chapter, including reconfiguration, microsystems, connectivity, etc.

Job-shops are a matter of interest, both in historical sense as a drive for flexible manufacturing as well as the relation to scheduling, specifically the job-shop problem. The job-shop problem is a variation of the classical Travelling Salesmen Problem (TSP). Where in the TSP problem the solution would be

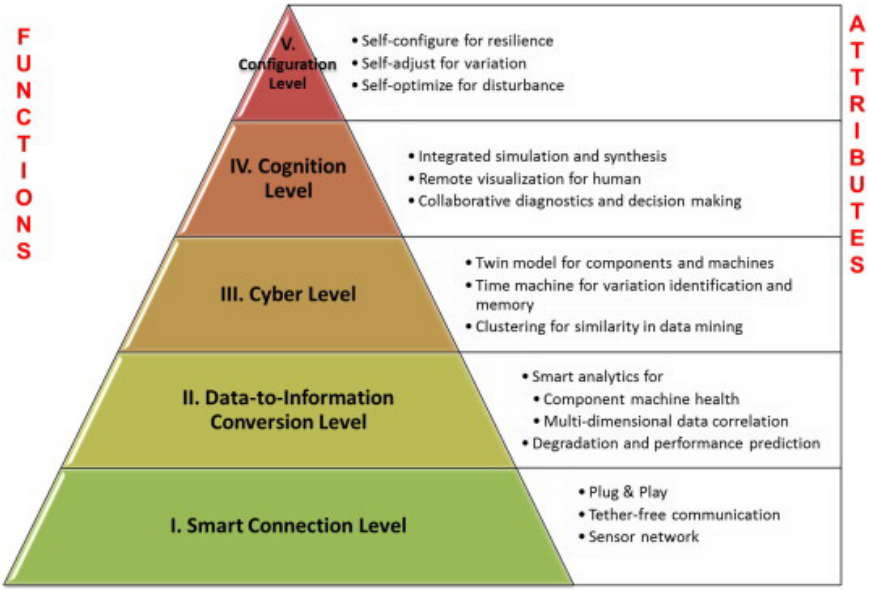


Figure 1.7: Architecture proposal of Cyber-physical Systems Source: (Lee et al., 2015).

the shortest route between a number of points (cities), the job-shop problem complicates the problem by introducing the limitation that a number of tasks (jobs) have to be performed in a specific order that can only be performed at a specific machine. The job-shop problem is therefore considered to be NP hard and as such has been extensively researched in the for (schedule) optimisation purposes, e.g. Applegate and Cook (1991). Of more interest is specifically the work of Bourne and Fox (1984), which offers five principles for the construction of systems to provide autonomous manufacturing. However, as Bourne mentions, at that time the (complete) automation of the general job-shop lies beyond the state of art.

Autonomic Computing is a field of interest since it addresses lowering complexity by managing technology with technology Computing (2003). Autonomic Computing was inspired by biology, where many processes are managed and monitored in an autonomic manner, e.g. the human nerve system keeps your temperature under control without a conscious effort. In IT systems this can be used by delegating certain tasks or processes to adapting policies that can take appropriate actions. These policies are based on a control loop that perform an autonomic capability. The IBM paper organises the control loops into four categories:

- Self-Configuring - The ability to drastically change in a changing environment, e.g. hardware configuration.

- Self-Healing - Being able to discover, diagnose and diagnose disruptions, e.g. detect system malfunctions.
- Self-Optimising - Automated optimisation to improve overall utilisation.
- Self-Protecting - Being able to anticipate and negate possible threats, e.g. a robot that stops a planned move to prevent a collision by detecting an object in its path.

1.2.2 Key Interests

From the state-of-the-art changes currently occurring in industry some specific fields are identified that require specific attention, as they could provide key interests to be able to increase flexibility and provide the ability to automate the manufacturing of high-mix, low-volume products with the use of state-of-the-art manufacturing systems:

1. Reconfigurability
2. Lowering Complexity
3. Autonomy

These three key points will be important aspects of the research and will be discussed all throughout this work. As mentioned before, logistics like supply management and scheduling will not be a key interest and will only be discussed if they are either required or directly affected. The scope and research focus will be discussed in more detail in the next chapter.

The overview of enablers, technological changes, and manufacturing paradigms shows that there are a lot of developments in state of the art manufacturing systems. For these changes to be brought to industry, it is important to experiment with these technologies, lower their complexity, and make parts more mature.

1.3 Thesis Overview

Figure 1.8 shows a first glance of what to come, but creating an overview of the thesis.

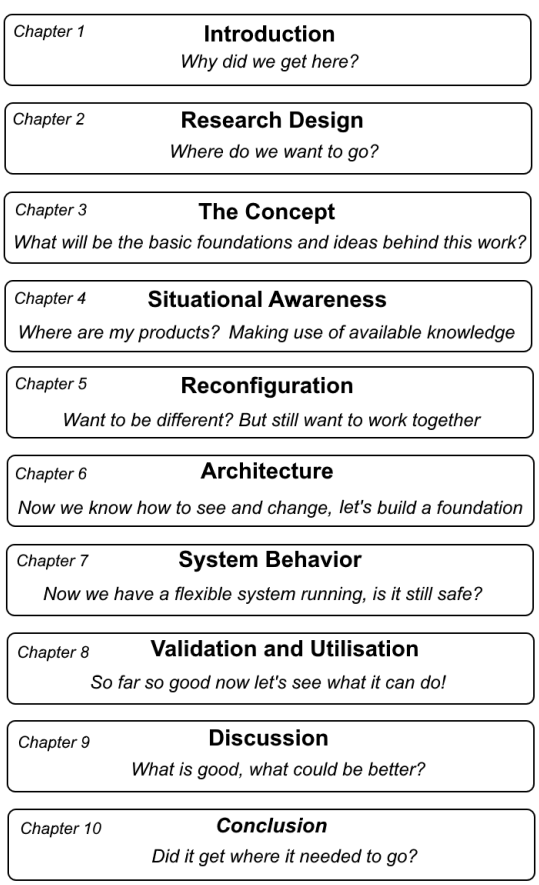


Figure 1.8: Overview of the thesis.

The Chapters can be split into 3 main parts:

Background:

Chapter 1 Introduction: Gives the background information and some motivation behind this research.

Chapter 2 Research Design: Discusses the research questions, methodology, and hypothesis.

Chapter 3 Concept: Describes the generic concept that is used, including the fundamental ideas and the requirements.

Main Chapters:

Chapter 4 Situational Awareness: Describes how to make smart use of available information to create a generic 2-step way for 6D localisation.

Chapter 5 Reconfiguration: Discusses how systems can be adapted and can still cooperate without prior knowledge of each other.

Chapter 6 Architecture: Investigates how a system architecture can be set up for use of grid manufacturing with autonomous and reconfigurable systems that need to both perform and be flexible.

Chapter 7 System Behaviour: Defines how the system is controlled and can be made safe.

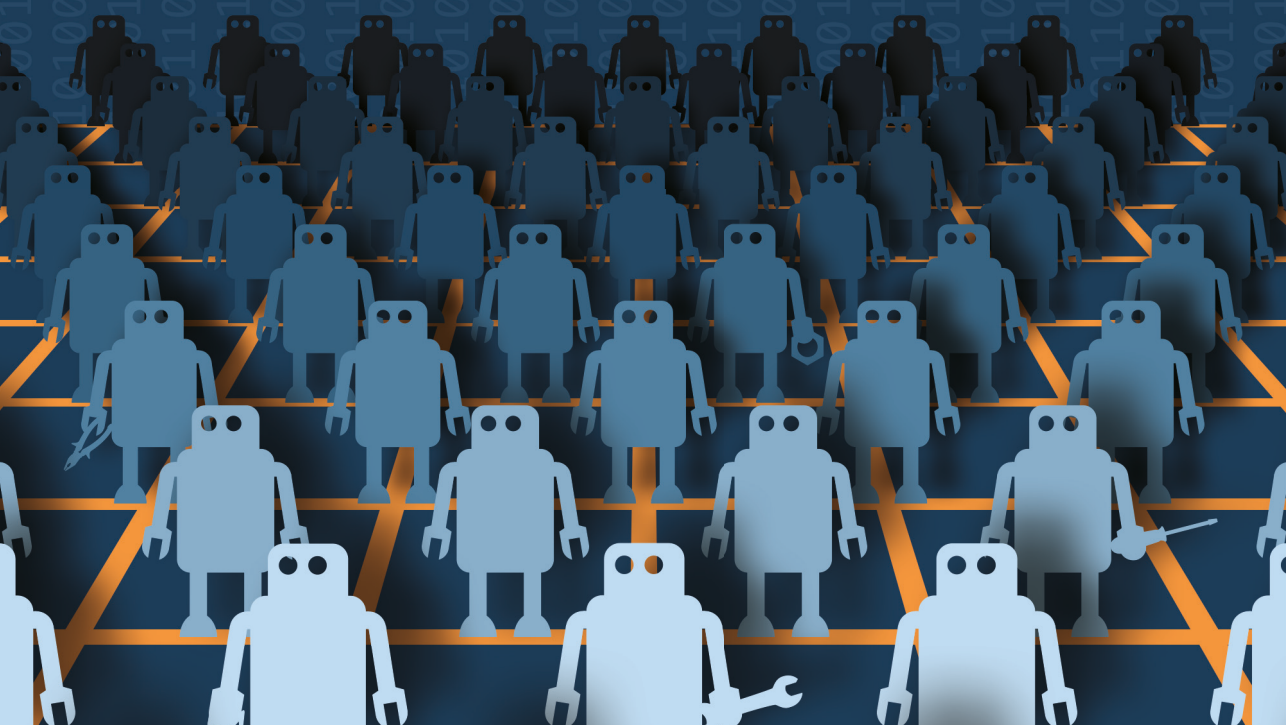
Chapter 8 Validation and Utilisation: Shows which opportunities are given by the developed systems and tests these in different cases.

Conclusion:

Chapter 9 Discussion: Evaluates the concepts that have been researched.

Chapter 10 Conclusion: Validates the work based on the original Research Design.

02



Research Design

“If we knew what it was we
were doing, it would not be
called research, would it?”

– Albert Einstein

Research Design

This thesis focuses on methods and concepts for flexible and agile manufacturing, i.e. the problems that arise when optimising for low-cost flexibility of autonomous manufacturing machines. In order to define a set of research questions we shall first discuss the problems and approach related to this work.

2.1 Research Approach

As observed in the introduction, technological advances and changes in industry are starting to have an increasing impact on the manufacturing industry. However, a true paradigm shift has not yet occurred, likely because the maturity and efficiency of these new technologies have yet to be proven. As Leitão states in the conclusion of his survey: 'The challenge is thus to develop innovative, agile and reconfigurable architectures for distributed manufacturing control systems, using emergent paradigms and technologies that can provide the answer to those requirements' (Leitão, 2009). Leitão also identifies some specific issues in this field, including:

1. The need for mature and proven technology - the majority uses laboratorial control applications without the need of physical devices (Hall et al., 2005).

2. Reconfigurability mechanisms - what architecture will support the society of distributed entities? (Leitão, 2009).

3. Development-related aspects - current platforms have limited scalability and robustness (Marik and McFarlane, 2005).

4. Prediction in disturbance handling systems - the integration of prediction mechanisms with identification and recovery of disturbances that can prevent these problems (Leitão, 2009).

The research in this thesis responds to these factors by creating concepts and an architecture that will include the hardware systems. This way a practical implementation will be made that shows concepts that could be feasible for industry.

2.2 Six Manufacturing Principles

The focus will be based on a new concept of flexible manufacturing platforms that produce products with a high mix and low volume. Six principles are identified:

1. A range of products can, in principle, be dynamically built on demand, i.e. the machines offer a (wide) range of *generic services*, which products can use in such a way that they can be manufactured as long as the required (generic) services are available.

2. Products can be *identified and localised dynamically*, i.e. the manufacturing systems should not need to be (re-)programmed to know and identify specific products; any new product can be added at any time into the production process.

3. Both products and manufacturing systems have no direct dependencies and can act *autonomously*.

4. Hardware should be *reconfigurable*, i.e. both hardware and software modules within a system should be able to be changed with limited downtime. In the case of the software, compiling code should not be required for a reconfigure action.

5. Machines should be *low-cost and single-purpose*, i.e. the flexibility that is offered should be possible with limited investment costs to guarantee experimental use and a valid business case.

6. System behaviour should be transparent and *safe*, i.e. the flexibility and dynamic behaviour of the system must be guaranteed not to lead to a high risk of use.

The six principles will lead to the ability to automate the creation of a range of products that can be built on demand, i.e. all products can be manufactured *ad hoc* as long as the parts and required services to assemble them are available in the manufacturing systems.

The principles also focus on limiting complexity by creating a minimum amount of interdependence between systems. The result is a concept that has been called 'grid manufacturing', where each manufacturing system delivers a service to a product. Since products and the manufacturing systems have their own purpose, i.e. the machine delivers a service, the product wishes to be produced, both autonomous and will work together dynamically. Hence, the system will not be a 'production line', since the need for services will depend on the specific product demand.

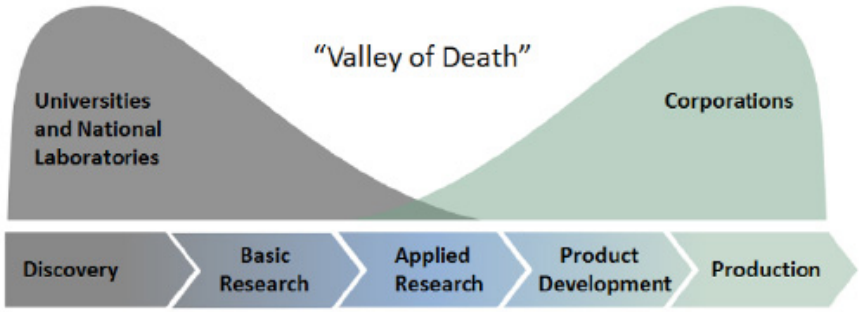


Figure 2.1: The Valley of Death, the gap between Research and (mature) product development, source: https://www.sri.com/blog/brazil-visits-sri-discuss-its-economic-development-roadmap - last accessed 23-12-2015.

Since the products will be manufactured dynamically without any specific programming it is important that the products are still produced safely and according to the specifications. The products will schedule themselves in negotiation with the manufacturing systems and, therefore, it is unknown which services are required. The flexibility makes it difficult to establish demand, and chances are that some manufacturing systems have a higher load than others because of this.

2.3 Problem Statement

While many paradigms and technologies show promise, they are not yet considered mainstream in industry. As mentioned before in Section 1.1, the industry is not yet adopting many technologies and is waiting for more maturity and proven technologies. The gap between the fundamental research of new technologies and the adoption as a mature technology in a business product is a widely known problem, which is called the 'Valley of Death' (Murphy and Edwards, 2003). Figure 2.1 shows the concept well by showing the amount of investments that are available at different stages.

The figure shows that commonly there are large investments for fundamental and basic research. However, while this generates new technologies they are commonly not sufficiently mature for corporations to be used. Therefore new technologies need to be proven and tested by applied research to gain the maturity that is required by industry. The applied research is not just used in the practical sense, but also generates new combinations of technologies, new concepts and experimental data that leads to new ideas.

The Valley of Death is a generic problem that needs to be taken into account. However, based on the introduction and design principles we identify

five specific problems:

Problem 1 Object Awareness - How can reconfigurable machines identify and handle a priori unknown objects?

Problem 2 Reconfigurable systems - It should be possible to quickly adapt the hardware and reconfigure the system by changing its hardware modules.

Problem 3 Architectural performance / intelligence gap - The platform should both be able to show 'intelligent' behaviour and have real-time performance to control the hardware.

Problem 4 Abstract Services - To use manufacturing as a service and limit complexity, the hardware cannot be 'known' by the product.

Problem 5 System Behaviour - While systems should be autonomous and modular, and work in a dynamic 'chaotic' environment, their behaviour should also be predictable and 'safe'.

Discussed in more detail: *The first problem*, **Object Awareness**, focuses on the always complex solutions to visually find, localise and inspect objects that are not known a priori. Especially in the context of reconfigurable manufacturing machines, where both the manufacturing machine itself can have changing properties, e.g. camera position or its changing working area. For computers it is inherently difficult to recognise objects in a dynamic, i.e. unknown, environment. Commonly computer vision is improved by controlling and standardising as many variables as possible. However, since reconfigurable manufacturing systems can be configured in many ways and objects are unknown in advance it is impossible to control all variables, e.g. positioning of the object, lighting, and position of the cameras can be different in any situation. Since the working area can also differ and objects are commonly 3D this makes it more and more complex to accurately visually analyse the objects in real-time. Objects should not just be identified by their shape (which is a 2D problem), but they can also be upside down, rotated or even on their side. Hence, to handle a product it is important for the equiplet to have a precise 6-dimensional measurement of the object by identifying it correctly, localising its x, y, z, rotation, pitch, and yaw positions.

The second problem focuses on the **reconfigurable aspect** of the systems. Since demand can change, it is important to adapt the systems to the (possibly new) demand. To create maximum flexibility, the system should be easy to adapt and if possible automatically update its use and services so they can become available to the products in the grid. A changing system is difficult

to control and use, since normally multiple manufacturing systems have to cooperate to create one product together.

The third problem is how can you create an **architecture** that combines the dynamic behaviour and flexibility for high-level functionality, i.e. understands its environment and cooperates with other systems in the grid, and low-level functionality, i.e. high-performance hardware control and algorithms. These different functionalities are based on different behaviour and therefore have different requirements. High-level functionality is based on abstract cognitive processes that use networked data and slow heuristic processes. Low-level processes are based on strict rule-based systems that have a direct (possibly real-time) impact on the actuators. As such they are usually written in native code using real-time systems. While native code could grant a higher performance it is also more difficult to develop, lowering flexibility and increasing complexity. Additionally, it is important that the high-level functionality will have no performance impact on the low-level systems.

The fourth problem focuses on the use of the manufacturing systems. The grid manufacturing concept asks for autonomous systems that provide generic services. Hence, the product is not aware of which manufacturing system will produce it beforehand. As a result, the product and the manufacturing system are not designed specifically for each other. For a product to be able to use the generic service that is offered they should be able to interface and understand each other. This asks for an ontology that both product and manufacturing system can use. The architecture should take into account which services and limits it can provide and match these to the requirements of the product.

The fifth problem focuses on the system behaviour. Since products are unknown and manufacturing hardware can be reconfigured there are many dynamic factors in a grid. Hence, it is difficult to define its exact behaviour. To be sure of the exact manufacturing specifications and safety aspects it is required to create specifications and procedures for actions that a hardware module can perform. A system should be created that defines the behaviour and describes how it will act during diverse situations like starting up/shutting down or errors. It is important that, even when dealing with new configurations or products, the machines can remain flexible in such a way that it can adapt to new situations, but at the same time be safe to use and not damage anything when performing these new actions.

2.4 Research Design Parameters

Based on the problems and principles, the Research Design Parameters can be identified and formulated (see below). Research Design Parameters are used here as presumptions that couple possible requirements for the solution to the

problem statement. These presumptions will be validated by experiments.

The main design parameter is that new (software) technologies, including reconfigurable and autonomous systems, could be used to create a new flexible manufacturing paradigm that is (cost-)efficient and predictable. The flexibility will have to provide for a much shorter time to market and an increased variety in products that can be manufactured in parallel. The challenge for the main design parameter is to keep the complexity (and therefore the practical applicability) of the smart and flexible approach under control. This has to be validated by developing a proof of concept that shows the abilities, performance and stability of such an 'agile' architecture. To assess the feasibility and practical implementation an experimental system will be fully developed, including low-cost hardware designed for this purpose. In more detail this brings us to the following five Design Parameters:

RP1 Machines can effectively provide generic services that can be used by different kind of products that are not known a priori, i.e. new products can be built on demand by machines never specifically designed for this product.

RP2 Computer vision is an important part of flexible manufacturing and can be simplified by making use of diverse data which is already available in the system.

RP3 The hardware of a machine can be reconfigured without the need of reprogramming the software.

RP4 The use of a simulator and transparent software control using standardised states can increase safety for reconfigurable manufacturing machines.

RP5 Cooperating agents in a grid in a non-hierarchical, i.e. heterarchical manner can be a flexible and efficient way to manufacture products in low quantities.

In more detail:

RP1 states that it would be possible to work with generic services, essentially making use of the expertise or capability of a system like you would make use of a human professional. Instead of designing a machine for one specific *a priori* set of actions it could perform anything within known boundaries. Much like a human professional, e.g. a carpenter who can build any product on demand, only limited by his competence and available material.

RP2 focuses on a specific aspect of flexibility, namely the ability of a reconfigurable machine to dynamically identify and interact with a priori unknown products.

RP3 states that the hardware of a system should be reconfigurable, without the use of a mechanic or engineer to reprogram the software. This also implies that not just the service is flexible, as mentioned in RH1, but the hardware itself should be flexible, in a sense that will become clear later.

RP4 The risks involved by dynamic manufacturing with autonomous reconfigurable manufacturing machines could be lowered with the use of simulation and a transparent state machine.

RP5 extends the idea of flexibility not only to a machine or service itself, but to a group of machines and how they are used. Basically this design parameter states that different autonomous machines could be managed dynamically in different ways to increase efficiency.

2.5 Research Questions

The problem statement and design parameters leads us to the main Research Question (RQ). RQ0: What could be the role of Reconfigurable Manufacturing Machines in the automation of high-mix, low-volume production?

This main question is split into five research questions:

RQ1 How can the detection and localisation of (previously unknown) objects be simplified and generalised?

RQ2 Can reconfigurable manufacturing systems be controlled without the need to reprogram them for every new product or hardware module?

RQ3 What options are available to combine flexibility and performance for software architecture in grid manufacturing?

RQ4 What risks are introduced due to the reconfigurable and dynamic behaviour and how can they be mitigated?

RQ5 What is the best way to utilise the possibilities of grid manufacturing and therewith validate its efficiency?

2.6 Research Methodology

The first step will be literature research, the second step to design a prototype architecture, and third to develop and test the architecture. For the third step a prototype architecture and several hardware and software systems will be developed to act as a proof of concept. This system will be the basis of future research and will be combined with several emulators to be tested in a large variety of cases. The study will combine quantitative and qualitative elements:

- Performance: quantitative research - based on empirical data using experiments from live proof of concepts and simulators.
- Abilities: qualitative research - comparing designs and architecture performance based on correlation.
- Processes: qualitative comparison - based on cases with (partly) quantitative data.

2.7 Research Objectives

The main Research Objectives (RO) describe how the research is achieved, i.e. which means will be used or created for the research. The central objective is to develop a proof of concept, based on the agile control architecture that maximises automated flexibility for high-mix, low-volume products. This objective is specified by the following (sub)objectives:

RO1 The concepts and requirements that are necessary to satisfy the design parameters: A flexible manufacturing system, which autonomously handles dynamic products on demand and which is reconfigurable and safe to use.

RO2 A robust generic approach for dynamically handling objects.

RO3 Investigate how machines can provide their services while being reconfigurable.

RO4 A software architecture that gives both the performance and flexibility required for the defined concepts, which gives the products and machines the means to effectively cooperate to dynamically build high-mix, low-volume products.

RO5 Predictable states to create safe use of reconfigurable machines in a dynamic environment.

RO6 Test the capabilities and efficiency of the system using simulation of multiple cases.

2.8 Scope

The thesis will highlight the system engineering and software aspects specifically, and will consider some hardware aspects when they are relevant for the overall concept.

The research intentionally tries to focus on specific items. Practical implementation is seen as an important aspect due to the fact that practical problems have a large impact on the maturity of technology and the adoption in industry. Therefore hardware and behavioural aspects are taken into account. However, some aspects are intentionally avoided when possible, e.g. scheduling, which even though it is an important field and has a high impact on the efficiency, scheduling is a field that has been researched to great depth. Therefore scheduling research is used in this thesis, but will only be discussed if it is of direct influence on the expected results that the research focuses on.

A specific limitation in this research is that the research is not meant for mass manufacturing means. Mass production will likely be more efficient with classic (less flexible) manufacturing means. Since these are designed specifically for high performance and throughput. This research focuses on opening automation to new markets where time to market and many different products in lower quantities are key.

To answer the research questions it is important to cover a variety of aspects, without losing focus on the original goal: creating even more flexibility for manufacturing systems. This will bring the research in a multidisciplinary area that connects with multiple fields, including: System Engineering, Embedded Systems, Intelligent Systems and Manufacturing.

2.9 Structure of the Thesis

Figure 2.2 gives an overview that shows where in the thesis the Research Design Parameters, Research Questions, and Objectives are discussed. Note that Chapter 3 does not contain any research questions as it primarily adds further specifications to the generic concept that was first introduced by Puik and Moergestel (2010) and discussed in the PhD thesis of Moergestel (2014).

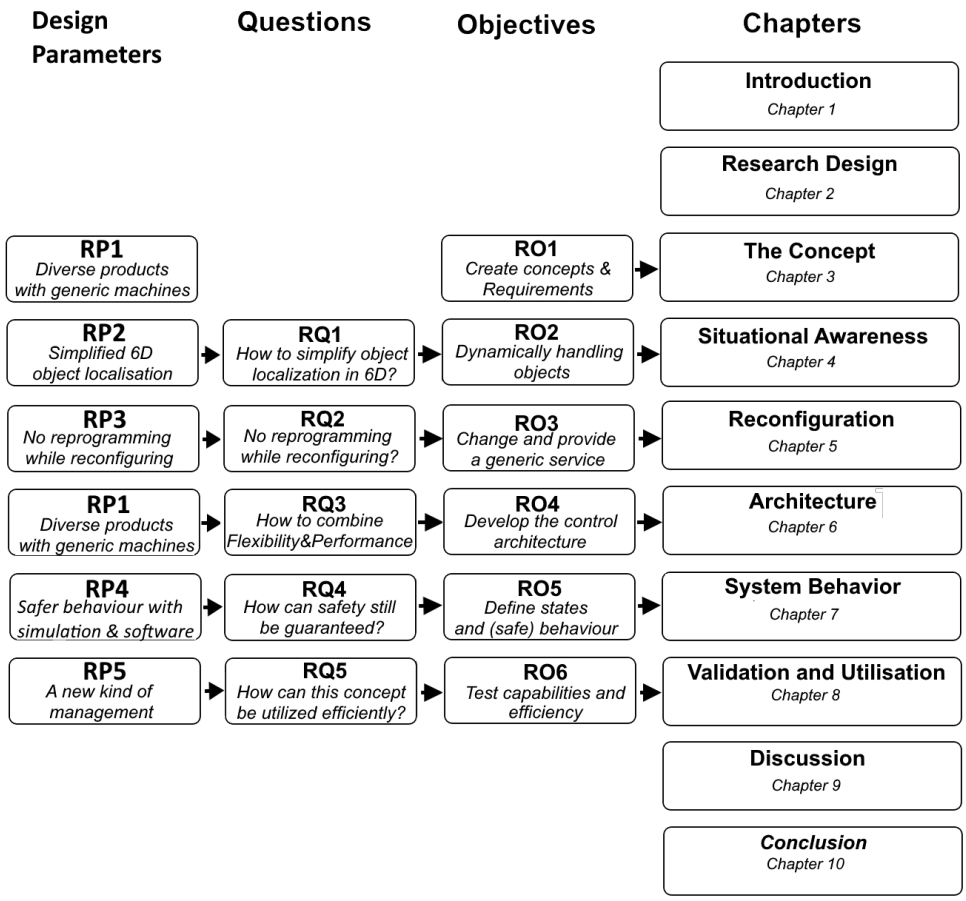
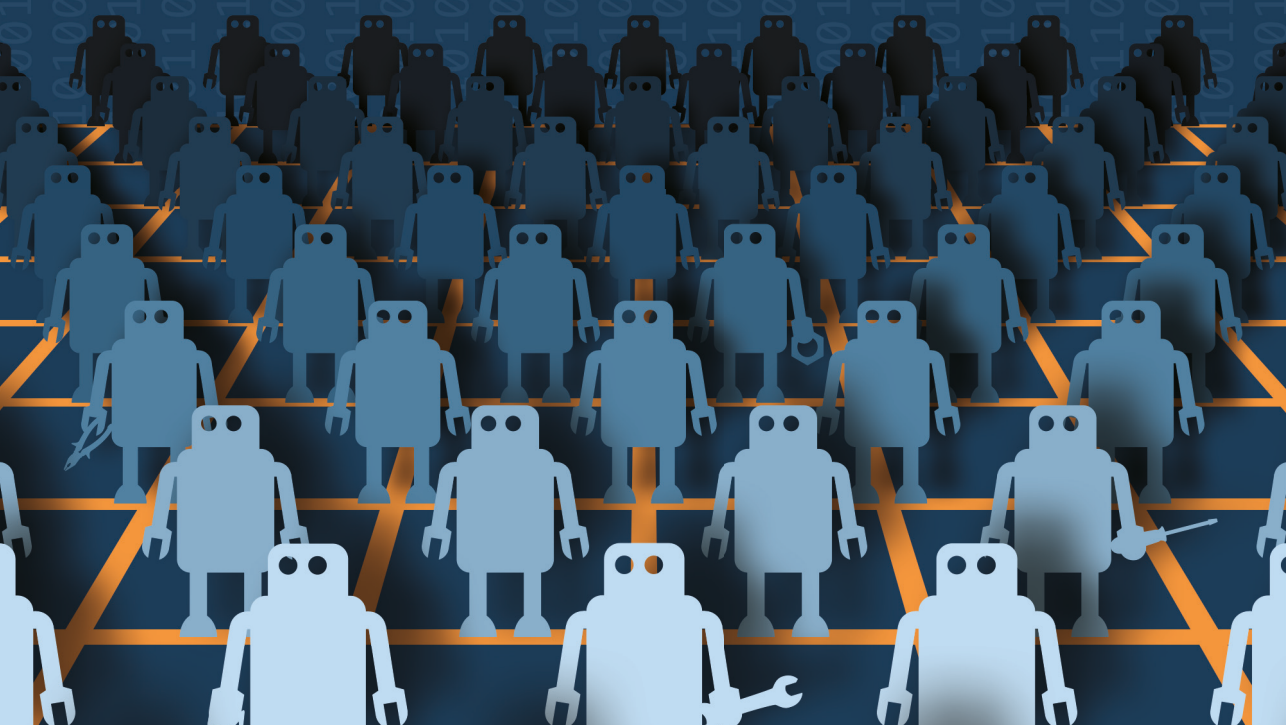


Figure 2.2: Overview of the thesis that connects the Chapters, with the research questions, research design parameters and objectives.

03



The Concept of Grid Manufacturing

“Religion is a culture of faith; science is a culture of doubt.”

— Richard Feynman

The Concept of Grid Manufacturing

Puik and Moergestel (2010) introduced two new terms, namely *production Grid* and *a grid of reconfigurable manufacturing systems*, the two terms evolved into a concept that has been called *Grid Manufacturing*. Grid Manufacturing is meant as a fully automated manufacturing approach, based on reconfigurable manufacturing machines that offer generic services that products can utilise dynamically. Part of this idea, the so called virtual 'product agent', which will be discussed in more detail later, was investigated by Moergestel (2014). This work expends these concepts by adding the so far unexplored concept of the reconfigurable manufacturing machines themselves. Parts of this chapter has been based on earlier work (Telgen et al., 2015b, 2012, 2015a).

3.1 Conceptual Idea of Grid Manufacturing

The main concept of Grid Manufacturing is based on the philosophy of Cyber-Physical Systems (Lee and Seshia, 2011). A grid will consist of three main types of systems.

- Reconfigurable Manufacturing Machines - The machines or robots that will actually conduct the assembly/manufacturing process.
- The product - Which is also seen as an inherent separate and autonomous system within Grid manufacturing.
- Logistic Systems - An example is a dynamic transport system with unmanned autonomous ground vehicles or multi parallel conveyor belts to transport systems dynamically, i.e. without predefined routes, between the manufacturing systems.

These three types of systems will be autonomous and can interact dynamically in a cooperating non-hierarchical manner. This is in line with the idea of Holonic Manufacturing (Höpf and Schaeffer, 1997). In general, both Cyber-Physical Systems and holonic ideas are partly based on abandoning of hierarchical control, moving towards autonomous, self-managed systems that mutually interact and cooperate. The philosophy behind Grid Manufacturing is also seen in different areas. For instance, (Laloux, 2015) mentions in his book 'Reinventing Organisations' that each period in time has its own organisational paradigm. In the last years we have been moving towards more self-management, and the concept of 'wholeness', i.e. striving not only for

yourself and the immediate professional needs of the situation, but also for others and aspects that are out of the immediate professional scope (such as durability, social practices). Advances that create connected computing platforms, like embedded systems and the Internet of Things (IoT), enable these human organisational paradigms to move into the technical realm. This development fundamentally changes the way machines and robots could be utilised. The move in current thinking could also directly impact business. Hence, the way we think of efficiency should evolve to encompass the ideas of 'wholeness', with a growing importance in flexibility and durability. This way of thinking is one of the fundamental thoughts behind Grid Manufacturing.

3.2 Grid Manufacturing

For the reconfigurable manufacturing systems a standardised hardware platform will be designed. It will consist of 'equiplets', as they can be easily equipped with a number of modules (Puik and Moergestel, 2010). Figure 3.1 shows a simplified example of the concept of a grid with autonomous systems. Equiplets provide their services based on their configuration to the grid server. A unique product is aware of its own design and request the first step in its manufacturing process, i.e. to print a number of 3D parts. The grid server replies with a list of possible equiplets that can provide a service like 3D printing. This prompts the product to negotiate with the capable equiplets and to find out if the equiplet can perform the service it requires within its requested specifications, which may include its needs in terms of schedule, quality, material, etc. The possible services that an equiplet can provide can change any time, based on either demand of the product or strategies that improve efficiency in the grid. In the shown example equiplet 1 mentions it might be able to perform the requested production step. However, it needs to reconfigures first. The product will keep negotiating with the equiplets in the grid until it finds a suitable solution.

Classic manufacturing is based on a Line Cell Module Device (LCMD) model as shown in Figure 3.2. The model represents a modular manufacturing process based on 4 hierarchical levels. The line is literally a 'manufacturing line' that is made up of a number of cells where a specific job is performed. A cell commonly uses multiple modules that perform specific actions, e.g. pick & place. The module can be decomposed even further into devices, e.g. a pick & place module will likely consist of several sensors and actuators, in which case each sensor and actuator can be seen as a device. The LCMD model is optimised for cost-efficient manufacturing of products that are made in high quantities. Since it is a linear model where products are made in a line, any change at any level will influence the entire manufacturing process. Hence,

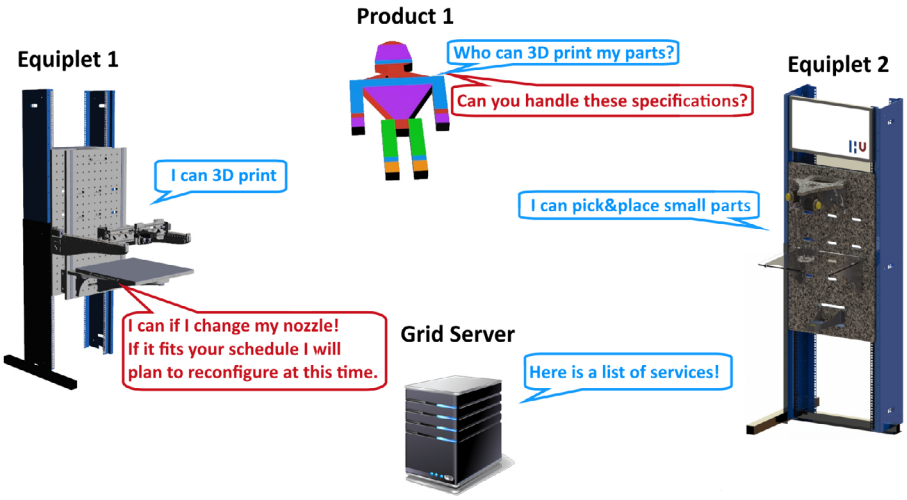


Figure 3.1: The simplified concept of grid manufacturing.

the product as well as the manufactured process will have to be matured completely before actual mass manufacturing can start.

The concept of grid manufacturing provides the opportunity to dynamically adapt to both product and equipment at any level, while the overall impact on production efficiency will be as limited as possible. This is performed by autonomous reconfigurable systems that provide generic services that products can use. All systems in the grid should cooperate to become self-organising. Because of the reconfigurable aspect of these systems, equiplets are not arranged in a line, but in a grid to emphasize that they can be used sequentially based on the current dynamic demand dynamic in the sense that different products can be made at any time using equiplets in any order using (soft) real-time negotiation and scheduling to plan how a possibly unique product will be manufactured. Figure 3.3 shows a rendering of an example grid with 12 equiplets, where every equiplet can have a different configuration to provide a variety of services that are required for the manufacturing process. Note that a grid does not require to have any specific form or size; depending on the demand they can be placed in any relative position based on the local logistic setup of the factory. It is imagined that a grid can contain a large number of equiplet, e.g. 100 equiplets that offer about 20 different generic services should be an option.

Equiplets provide capabilities that are based on the configuration of the specific modules that are installed. In the grid manufacturing context, reconfiguration is defined as adding/removing/changing modules within the equiplet

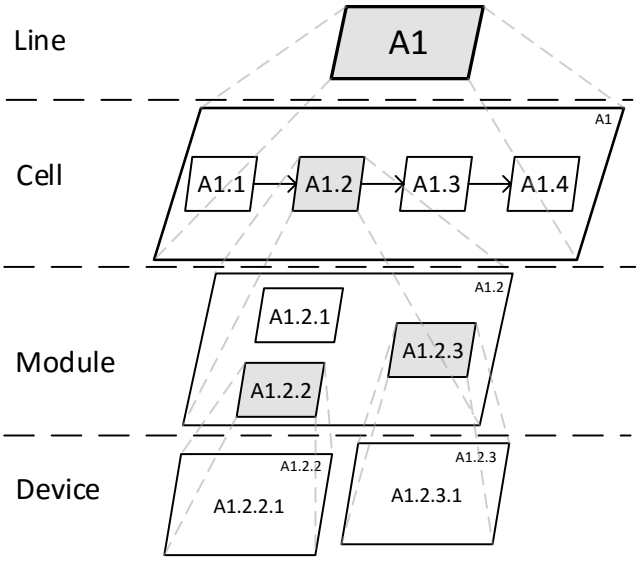


Figure 3.2: Classic Line Cell Module Device (LCMD) structure.

so as to change its capabilities. Reconfiguration includes both the physical change as the adaptation and configuration of the software to control the equiplet.

In contrast to LCMD, the architecture for Grid Manufacturing is called the Grid Equiplet Module (GEM) Architecture, as shown in Figure 3.4. With GEM the systems are loosely coupled, the Grid layer provides services to the autonomous equiplets. Modules are commonly designed as Components off the Shelf (COTS).

Olhager (2010) describes the Customer Order Decoupling Point (COPD), which is defined as the point in the value chain, where the product can still be linked to a customer order. These can be split in four categories: (1) make-to-stock (MTS), (2) Assemble-to-order (ATO), (3) make-to-order (MTO), and (4) engineer-to-order (ETO). From the Product perspective, Grid Manufacturing can immediately manufacture a range of numbers, as long as the, capacity, capability, and parts are available. Hence, this would be typical as assemble-to-order.

Besides delivering flexibility, the concept also introduces a manner to bring the product designers and production experts closer together. In the past, lines were made specifically for one product, and as such it would come at a high cost to take a working line offline to create a prototype for a new product. Grids can dynamically handle various products in parallel, such that a product designer is able to use the same manufacturing equipment that is used for the

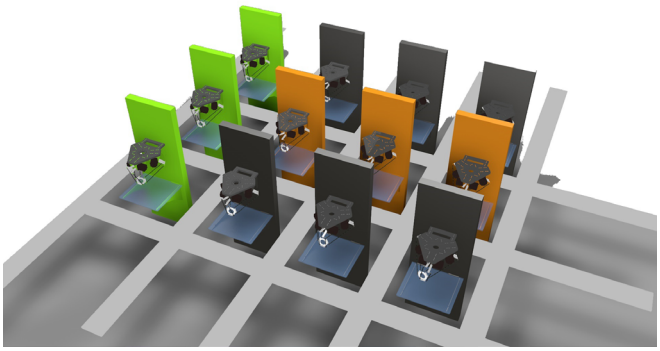


Figure 3.3: Example of a small Grid Structure.

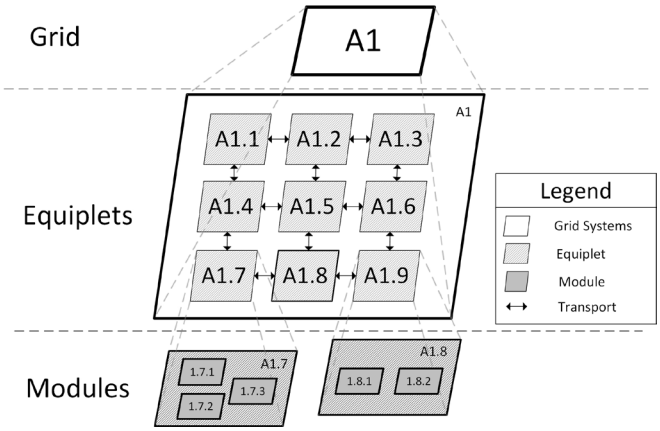


Figure 3.4: GEM Architecture.

final manufacturing to create prototypes and test the production phase. This shortens the time-to-market and lowers costs.

To further work out the required platform and analyse the success of this concept the requirements should first be discussed.

3.3 Requirements

As stated in the research objectives section 2.7, the main objective is: "To develop a proof of concept, based on the agile control architecture that maximises (automated) flexibility for high-mix, low-volume products". The proof of concept will be used to answer the research questions.

The requirements will be the basis for the grid manufacturing architecture and the proof of concept. Since the design of complex systems as used in

grid manufacturing is challenging, the requirements are loosely based on the Axiomatic Design methodology developed by MIT (Suh, 2001). Axiomatic design uses design principles or axioms, i.e. premises or starting points for reasoning. In Axiomatic design the characteristics and needs are translated into four domains.

- Customer Domain - Customer Attributes (CA) - The customer's needs.
- Functional Domain - Functional Requirements (FR) - What it needs to do.
- Physical Domain - Design Parameters (DP) - How can it be made.
- Process Domain - Process Variables (PV) - The variables that characterise the design in the process domain.

The Physical Domain and the Process Domain will not be discussed at this time, since the high amount of required detail will not contribute to the concept described in this chapter. However, it will be discussed in separate chapters when it directly contributes to the research.

The requirements will be split into three different subsections. First, the motivation behind the concept and its requirements will be discussed; second, the customer attributes; and third, the functional requirements.

3.3.1 Motivation

Successful launch of new and unique products will lead to competitive advantages, which are at the heart of a firms performance in competitive markets; this statement made by Puik (2016) is one of the main drives behind Grid Manufacturing, which is designed to optimise flexibility and be able to bring new products to the market. His statement was based on Porter and Advantage (1985): "Introducing a significant technological innovation can allow a firm to lower cost and enhance differentiation at the same time, and perhaps achieve both strategies. Introducing new automated manufacturing technologies can have this effect." Koren (2006) gives a similar message by discussing that high production efficiency and rapid response to changing customer demand are dominant conditions for enterprises to stay successful. All these remarks emphasise the need for rapid automated adaptability.

The global market is influenced by increasing purchase power and globalisation. Hence, to stay competitive it is essential to be able to quickly industrialise a product. This is also the message of the PhD thesis Manuscript of Puik (2016). He discusses that being able to bring a product to the market quickly is an important aspect of creating sufficient market penetration, which will also extend the product life cycle This is shown in Figure 3.5, if

the introduction of the new market is delayed this would result in a loss of turnover.

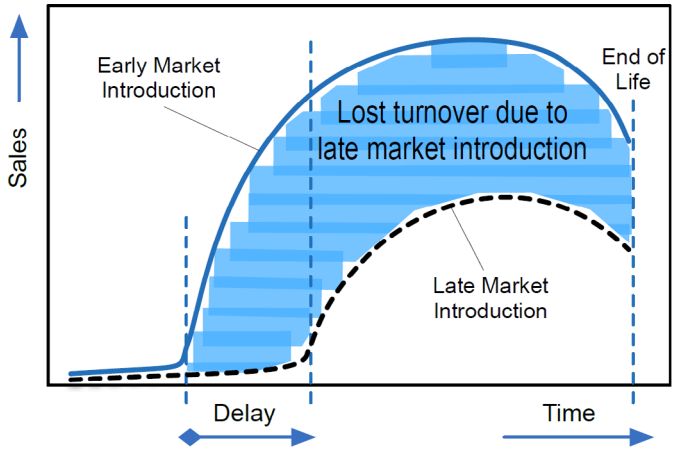


Figure 3.5: Delayed market introduction reduces sales and duration of sales of products; total turnover will be substantially lower. (Puik, 2016)

Puik mentions four important aspects, three of which are directly connected to Grid Manufacturing: (1) The pressure on lead time, i.e. the total time required to manufacture an item (production, queue, inspection, etc.), is leading to a change in the production process. Manufacturing equipment should be 'agile'; in other words, it should be able to adapt quickly and as such be able to bring a new product to market quickly. This is the area of Reconfigurable Manufacturing Systems (Koren, 2006; Mehrabi et al., 2000; Gunasekaran, 1999; ElMaraghy, 2005; Puik et al., 2013b). This view is also supported by the survey of agent-based distributed manufacturing control by Leitão (2009), who not only confirms the need for innovative, agile and reconfigurable architectures, but also mentions that industry is slow to adopt these new technologies. This is mainly because of the focus on fundamental properties, like scheduling and efficiency. However, the focus should become practical application and integration of systems in such a way that they can be applied directly in industry (Shen et al., 2006). (2) The development of new product freezes the resources of a company, which are then only released gradually after market introduction. This slows down the company in a fast-changing market. Hence, there is the need for (i) quicker market introduction, (ii) more efficient product development, and (iii) effective manufacturing engineering followed by pilot production and ramp-up to required production numbers (Moore, 1998). (3) Based on Onori and Barata Oliveira (2010) experience in the European

EUPASS project (Evolvable Ultra-Precision Assembly Systems) Puik mentions the difficult relation between 'Product Development and Manufacturing Engineering. This is primarily due to the fact that processes being catered for application in the assembly process, systems are insufficiently documented and structured. Hence, a 'Design for Assembly' strategy could be applied that incorporates designers and production engineers to work together on the product design. However, in reality these people work at different locations and communication flaws are common in the industrialisation process. Therefore, grid manufacturing will bring the industrialisation right at the doorstep of the designer, by being able to experiment with the manufacturing process itself, and with limited to no influence to overall production capacity of the grid.

3.3.2 Customer Attributes

The Customer Attributes describe the needs of the customer. In this case the customer is a manufacturing company that wants to automate the manufacturing of high-mix, low volume products.

Based on the motivation, the following aspects are identified as being important for the Grid Manufacturing proof of concept:

- A modular and reconfigurable design - to be able to automatically assemble-to-order;
- Low-cost - to be able to be competitive on the market;
- A practical 'applied' implementation - such that parts of the proof of concept can be applied to industry in the short term;
- The part/product transport system should be flexible - to handle differentiated automated assemble-to-order products;
- Machines should deliver generic services - as well as handle differentiated automated assemble-to-order products.

While also of interest, human interaction within grid manufacturing is placed out of the scope for this thesis.

3.3.3 Functional Requirements

The functional requirements in Axiomatic Design are given by answering the question, 'what should the system do?' This is placed in the scope of all software systems for multiple autonomous reconfigurable manufacturing machines. At this time only the high-level functional requirements that are typical for grid manufacturing are mentioned. Some of these requirements will be used

for decomposition to more specific requirements in later chapters. Note that the developed software for Grid Manufacturing consists of more than 2500 code source files and as such are only discussed in this Chapter on a higher abstracted level.

The high-level conceptual functional requirements are clustered to a three-level decomposition for the manufacturing systems in the grid and a fourth for the product entity that represents the product:

Grid A decentralised system where equiplets and products cooperate.

Equiplet An autonomous, modular, reconfigurable, single-service, low-cost manufacturing machine.

Module A hardware module that provides one specific function within an equiplet.

Product The entity that will represent the product.

Grid Level

The grid level offers a number of services that support the equiplets and possible logistic systems. The grid in general has no hierarchical control of its underlying systems (the equiplets, products, etc.), since this would limit its flexibility and complexity. Hence, the functional requirements of the Grid explicitly mentions things it should 'not' do, like adapting, i.e. reconfiguring equiplets or adding or removing equiplets entirely without interference to the overall grid. To enable this change and create the ability to dynamically manufacture differentiated products on order it is required to have a transportation system that can dynamically create a custom path between the equiplets. Hence, there are four Functional Requirements for the grid level.

The grid should be able:

GFR1. To offer generic services to a variety of differentiated products.

GFR2. To validate and assess its own efficiency.

GFR3. To adapt (remove or add) services/equiplets with limited to no interference to other products.

GFR4. To provide for product transport dynamically between each equiplet.

What is required for GFR1, and GFR3 is discussed in more detail in Chapter 5. Also, GFR2 is used in Chapter 8, where reconfiguration is initiated based on the overall efficiency, i.e. utilisation, of the grid. GFR4 is stated as a condition, but is not discussed in detail since there are several options available

to perform this, e.g. in the related field of order picking (De Koster et al., 2007), which could be applied for this cause. Or, more directly connected to Grid Manufacturing, the work of Moergestel et al. (2015).

Equiplet Level

The equiplet level focuses on the reconfigurable manufacturing equipment, the equiplets. As explained before, the concept of the equiplet is that they are autonomous, reconfigurable and provide generic services. However, since the reconfigurable and dynamic aspects also introduce risks it is important that the equiplets are safe to use. Efficiency is always a factor to be cost-effective.

There are seven Equiplet Function Requirements (EFR). An equiplet should be able to:

EFR1. Provide a specific 'generic' service to a product.

EFR2. Be reconfigured (adding or removing of modules - to provide a different service).

EFR3. Work autonomously, i.e. it has no strict dependencies with other equiplets or does not create interferences for other equiplets.

EFR4. Automatically adapt its control software when modules are added or removed.

EFR4A. Let its new service (capability) be known to the grid.

EFR4B. Update its system behaviour and safety software.

EFR5. Translate abstract instruction from a product and translate it to instructions for its own specific hardware modules.

EFR6. Efficiently control the hardware in real-time.

EFR7. Identify and localise, i.e. define the position and orientation of *a priori* undefined objects, e.g. parts or products.

EFR1, EFR2, EFR4, and EFR5 are used as input for Chapter 5. The safety aspect EFR4 is also discussed in Chapter 7. EFR3 and EFR6 for Chapter 6, and EFR7 is discussed in Chapter 4.

Module Level

The modules are the reconfigurable parts of an equiplet. To be automatically reconfigurable, or, as it is called in autonomic computing, self-configurable Computing (2003), it should be known what the characteristics are in such a

way that the equiplet can adapt its properties, be safe, control the module, and use the module effectively.

There are two Module Functional Requirements (MFR). A module should be able to:

MFR1. Know its own characteristics.

MFR2. Accept and perform instructions from the equiplet.

MFR1 and MFR2 are both discussed in more detail in Chapter 6, where several modules are discussed, in Chapter 5 where is discussed how they are controlled, and Chapter 7 concerning how the characteristics are used for safety aspects.

Product Representation Entity

There are seven Product Representation Functional Requirements (PFR) during the manufacturing phase. Note that the Product entity is the main topic in the related work of Moergestel (2014) and therefore placed mainly out of scope for this thesis. However, it is discussed when it interfaces with the manufacturing systems or when the functionality of the product entity needs to be adapted because of the new insights given in this thesis.

The Product Representation Entity should be able to:

PFR1. Coordinate its own production.

PFR2. Know which parts it requires to be completed.

PFR3. Know which (abstract) services it requires to be assembled (production steps).

PFR4. Determine which services are available.

PFR5. Communicate with equiplets to determine if they can perform a production step.

PFR6. Create a (viable) schedule on how it will be produced.

PFR7. Log its production/assembly history.

3.4 Fundamental Technologies

Based on the concept and the proposed requirements, two technologies will be fundamental for the concept of Grid Manufacturing: Agent Technology and Cyber Physical Systems. These are discussed in the following subsections.

3.4.1 Agent Technology in Manufacturing

(Paolucci and Sacile, 2005) noted that they can create a flexible, scalable and reliable production system using agents. Hence, agents fall within the philosophy of self-managed autonomous systems that is part of the design philosophy of Grid Manufacturing. (Moergestel, 2014) also uses agents to represent products. However, the work by van Moergestel mainly focuses on the product agent and leaves many questions on the perspective of the manufacturing machines.

Since agents are an important aspect of Grid Manufacturing more detailed information about agents will be given. We distinguish two types:

1. Reactive agents
2. Reasoning agents

Figure 3.6 shows an example of a standard reactive agent cycle that perceives its environment through sensors, interprets it according to standard rules, and chooses an action accordingly and acts using actuators to change something in the environment.

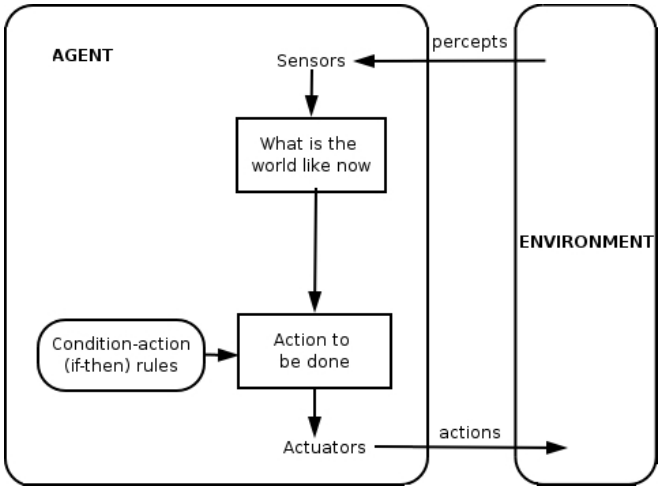


Figure 3.6: Simple reflex of an Intelligent Agent.

Reasoning agents exist in multiple types, the best known of them is the belief-desire-intention (BDI) agent, see Figure 3.7. The BDI agent uses the philosophy of (Dennett, 1987) and (Bratman, 1987). The BDI agent uses its sensors to build a set of beliefs, where its desires are a set of accomplishments that the agent wants to achieve. The BDI agent can choose desires that it wants to actively try to achieve; these are its goals. It then commits to a goal

to make it into an intention, activating a plan that consists of actions that it will take to achieve its goal and thus satisfying its desire. Ideally, a BDI agent uses the following sequence to achieve this (Rao et al., 1995):

1. initialize-state
2. repeat
   (a) options: option-generator(event-queue)
   (b) selected-options: deliberate(options)
   (c) update-intentions(selected-options)
   (d) execute()
   (e) get-new-external-events()
   (f) drop-unsuccessful-attitudes()
   (g) drop-impossible-attitudes()
3. end repeat.

As shown in the sequence the agent adapts its behaviour by actively deliberating and updating its intentions based on external events.

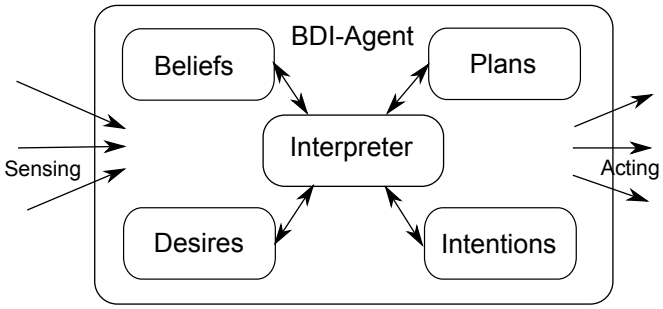


Figure 3.7: Beliefs Desire Intention Agent, adopted from (Wooldridge, 2009).

Figure 3.8 shows the concept of Multi Agent Systems (MAS), where multiple environments communicate and work within an environment. Agents interact and can cooperate or negotiate to achieve common goals. Any agent can have a specific role within a MAS and can interact with the other agents using specific permissions and responsibilities.

MAS can also be associated with **Environment Programming**. Environment Programming is seen as an abstraction where the environment is seen from the agent's perspective. Objects in the environment that the agent interacts with are seen as programming models and are named 'artefacts'. This

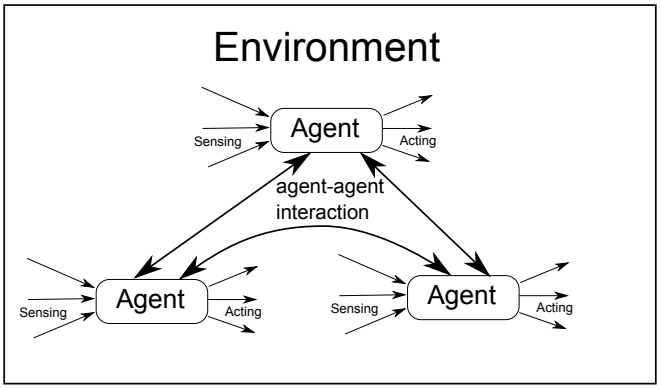


Figure 3.8: Multiple agents forming a Multi Agent System (MAS).

creates an extra abstraction where objects keep their abstraction layer and can be used effectively by the agents (Ricci et al., 2010).

Since Multi Agent Systems fall right within the specifications and philosophy for Grid Manufacturing they will be used into the design of the Grid Manufacturing Control Architecture, which will be called: Reconfigurable EQuipletS Operating System (REXOS where the QS is replaced by an X).

3.4.2 Cyber-Physical Systems

Cyber-Physical Systems are the combination of embedded systems, that use microsystems, i.e. sensors and actuators that are connected through the internet of things. An aspect of Cyber-Physical Systems is that the 'cyber-part' has a model of the physical world that it uses to control the actuators. This is basically the next step of classic sense-plan-act methodology applied to a broader range of connected devices.

An important aspect of the REXOS system is the Cyber-Physical aspect. The concept is directly applied to the REXOS system and also fits in well with the view of 'agents'. There, every Cyber-Physical entity is represented by an autonomous agent. Hence, REXOS implements two main agent types:

- The Product Agent - representing the product.
- The Equiplet Agent - representing a reconfigurable manufacturing system.

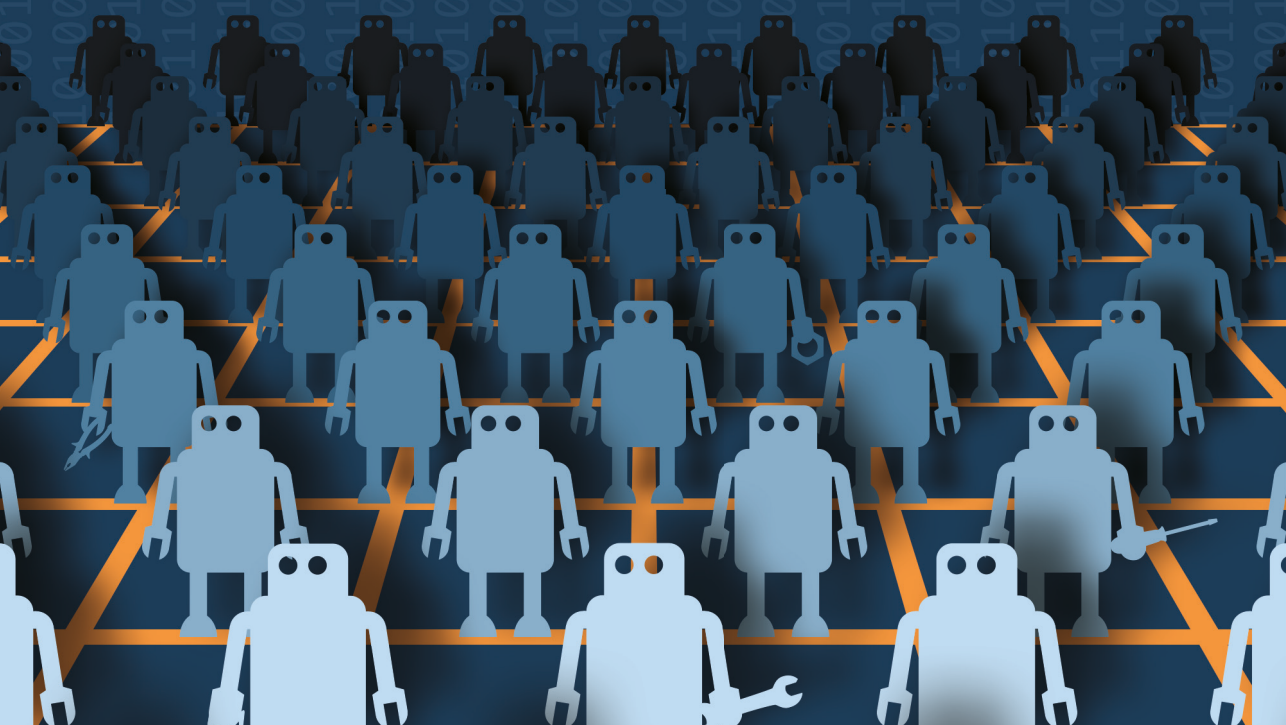
These agents were originally introduced by (van Moergestel et al., 2011), Van Moergestel focused on the creation and the development of the Product Agent which has been presented in the PhD thesis (Moergestel, 2014). In his work, the equiplet agent was primarily used to provide scheduling information

for the equiplets. This thesis will expand on that work, by extending it with the perspective of the Reconfigurable Manufacturing Systems. The research will show (1) how to control the hardware, (2) create flexibility and (3) how to add reconfiguration. This will be discussed in more details in Chapter 5.

3.5 Conclusion

The Concept Chapter demonstrates how this thesis looks at manufacturing; products and machines are not seen as passive objects, but as autonomous proactive entities, i.e. Cyber-Physical Systems. The chapter highlights the ideas behind grid manufacturing and is seen as a more detailed introduction to the research that will be discussed. It shows the concepts of reconfigurable manufacturing systems, the equiplets, and introduces agent technology and the philosophy behind Grid Manufacturing. It also lays the foundation for the research by providing the functional requirements that would make this possible. On top of the research questions, the requirements give practical guidance for the objectives and development of the platform that will form the basis for the rest of the research.

04



Object Awareness

“Everything must be made as simple as possible. But not simpler.”

– Albert Einstein

Object Awareness

Object awareness is one of the aspects of grid manufacturing, or any other paradigm where objects need to be handled dynamically in an unknown environment. Object awareness in this context can be seen as the ability of a machine to be aware of the objects that are within its 'working area', i.e. the space wherein the machine can move and/or interact with other objects. This requires not only to see if an object exists within its working area, but also to know what the object is and to determine its position and orientation, so that it is able to interact with it.

In most manufacturing systems the ability to handle parts is commonly performed by feeder systems or conveyor belts. However, in grid manufacturing all systems are dynamic, and the use of standard feeders or belts that would orientate specific parts to an exact location would limit flexibility. As a result, it is not known how an object is orientated when it reaches its expected location.

Grid Manufacturing extends the idea of Cyber-Physical Systems (CPS). This is done by introducing various smart (self-managed) entities, i.e. agents (Puik and Moergestel, 2010), that coordinate various processes within the grid. The agents will be able to autonomously handle processes such as the logistics, reconfiguration, production. Since these agents have intelligent properties, they contain information that could be useful for various purposes, and as such it would be of interest to examine how their knowledge provides opportunities for the identification and localisation process that is required to handle objects dynamically.

4.1 Simplifying Object Recognition

4.1.1 Problem Description

Since products and manufacturing equipment are decoupled in design, the location and positioning of the product and the vision equipment cannot be known a priori. Hence, it becomes necessary to be able to identify the objects using real-time sensor data and make use of live configuration data. This creates a number of complicating factors that need to be mitigated:

1. The hardware configuration can be variable and needs to adapt to the current machine setup;

2. Any *a priori* unknown object that enters the grid needs to be recognised;

3. The machine needs to verify that the product has been scheduled is actually within its working area;
4. The positioning, i.e. place and orientation of the product in relation to the machine, needs to be determined in real time.

As an extra complication it is also necessary to determine the objects position in 6 dimensions, which is called 6D localisation, see figure 4.1.

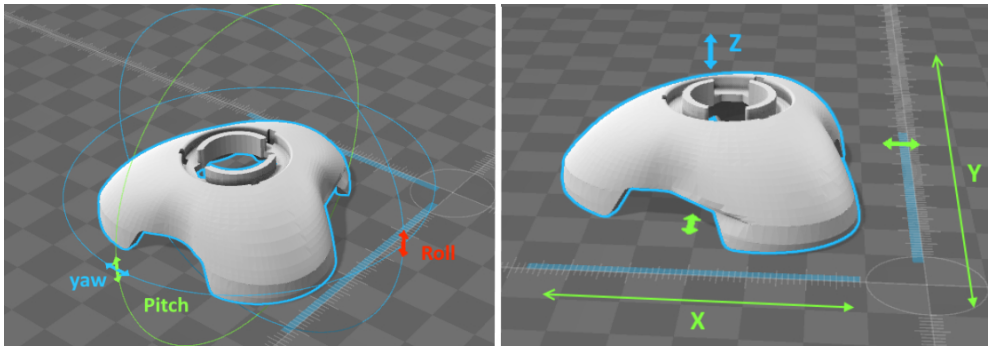


Figure 4.1: The 6-Dimensional orientation of an object.

Precise measurements are required for the position axis: X, Y, Z, and the orientation axis: Roll, Pitch, and Yaw, to correctly handle an object. This task could be performed by stereo or 3D cameras that inspect an object from multiple angles using computer vision techniques. However, vision is commonly implemented using custom code that is specifically tailored for a known object. Hence, it would be preferable to find a more flexible and low-cost option that handles objects within the working area of a reconfigurable manufacturing machine.

4.1.2 Requirements

As mentioned in Section 3.3.3, EFR7, the equiplet should be able to identify and localize, i.e. define the position and orientation of a priori undefined objects, e.g. parts or products. This chapter will provide a proof of concept that follows from the problem description and fulfils requirement EFR7. An object awareness module will be developed for this purpose. For the design of the module, EFR7 can be decomposed in the following sub Functional Requirements:

EFR7a to identify which *(a priori* unknown) objects are within its working area;

EFR7b to determine the location (in 6D) of the object.

4.1.3 Additional Research Questions

The main question for this chapter, as mentioned in Chapter 2, is: "RQ1. How can the detection and localisation of (previously unknown) objects be simplified and generalised?" Within the context of Grid Manufacturing this leads to two additional Research Questions which will be discussed in the current chapter:

RQ1a What data is available and how can the data that both product and machine contain be effectively combined to help the identification process?

RQ1b Can the 6D localisation effectively be performed by 2D camera systems?

4.1.4 Research Design Parameters

The main Research Design Parameter is that modern distributed systems have a large amount of (meta)data that could be effectively combined to be used together with data from cameras to identify and localise objects. This follows from the following Research Parameters (RP), which in turn follow from the sub research questions:

RP3a In the concept of Grid Manufacturing both product and the machine have a distributed Cyber-Physical entity, i.e. an agent, that acts as a representative of the system. The agent contains a large amount of data of the system it represents, and as such it is possible to use the (meta)data dynamically in a running system to predict specific characteristics and combine knowledge to simplify the identification and localisation of objects within the working space of a machine.

RP3b When making use of the metadata mentioned in RH1 it should be possible to combine this with the data of 2D cameras to create a standardised process to localise an object.

4.1.5 Research Objectives

The objective of the current chapter is to investigate whether it is possible to create a standardised process that combines knowledge of the different entities, i.e. products and reconfigurable machines, in real time to simplify the identification and 6D localisation problem for *a priori* unknown products. This will be performed using the following steps:

RO2a Identify the available (meta)data that can be used in relation to enhance the vision process.

RO2b Design a process that uses the (meta)data to identify an object.

RO2c Use the knowledge of the identified object to create a (simple) standardised localisation process.

RO2d Test the process with verifiable results.

RO2e Determine the precision and practical appliance of the process.

4.1.6 Literature Overview

While there are systems that are known to localise objects with the use of precise measuring equipment, the challenge is to improve the process using low-cost, reconfigurable components, in line with the concept of Grid Manufacturing. The concept requires a new generic process that makes use of the knowledge that exists of the system. This could be done by automatically setting up the vision system which is also able to detect new objects, i.e. products that were never seen and detected before and have now been added for production within the grid.

In classic manufacturing systems the orientation problem of objects is commonly used by using a feeder system that automatically orientates parts in the preferred way. (Stappen et al., 2002) shows a number of ways to do this using a range of sensorless manipulations. However, since this requires the development of a specific feeder system per part this is not appropriate for the flexible approach required in Grid Manufacturing. Hence, the focus should be on more dynamic approaches that do use sensors to localise and manipulate an object.

Flynn and Jain (1989) directly uses CAD based models to match objects. However, their approach focuses on performing geometric inferencing to obtain a relational graph representation of the object. The representation is then stored in a database and used for object recognition. However, this way the recognition is only effective for matching between a set of objects, but does not solve the 6D localisation problem.

Azad et al. (2007) uses a different approach; in this paper a stereo vision camera is used for 6D object localisation in order to grasp objects with a robot. The approach currently seems limited to cylindrical, textured objects. The system detects highly textured points, calculates the 2D contour, then uses the stereo vision to calculate a 3D point and fits it into a 3D plane. By using a 6D pose the object is then placed into the world coordinate system. In this paper the model is generated manually, while it does make some interesting points, it does not make use of possible meta-information. The work is more focused on determining the shapes of objects to be picked up. Using meta-information would simplify the problem, making it more robust and easier to implement.

Lowe (1987) shows a general framework for visual recognition. It uses 2D images to recognise 3D objects without the use of depth information. This also uses the 3D object models to verify the image features and makes use of prior knowledge of objects.

Kallmann and Thalmann (1999) focuses on interaction between (virtual human) agents and objects. The paper makes use of the 'smart object' concept. A smart object is a modelled object that has its own interactive features. These possible interactions are stored in the object as well. This is remotely similar to the 'agent' approach of Grid Manufacturing. However, in contrast to a smart object, an agent will also initiate (proactive) action based on its own behaviours.

4.2 Metadata in the Grid

The main idea of on the concept is that two main agents can be expected to hold data that could be useful for the vision process. The two agents are the equiplet agent and the product agent. The equiplet should have precise knowledge of its configuration. This includes a Computer-Aided Design (CAD) model of itself with a precise position of its working area. In line with the Cyber-Physical way of thinking, the product agent should also have a model of its own design. At this moment three kinds of metadata is identified to be of interest:

- Position of the Working area.
- CAD Model of the product.
- Position of the camera(s).

A standard equiplet is usually configured with a 'workplane', see figure 4.2.

The workplane is a flat surface on which an object that needs to be handled is positioned. Hence, it can be assumed that all objects that need to be handled by an equiplet are placed on top of the workplane or on a tray that is positioned on the workplane. This provides specific information about the position in the Z axis of the object. An object in rest always has a limited set of possibilities for its orientation. This is made clear in Figure 4.3.

This object is currently not stable on a flat plane and will have only 2 possibilities for the pitch and yaw dimensions. Knowing this greatly simplifies the localisation process.

The localisation data together with the model of the product can also determine where it could be handled, i.e. gripped to be moved. The second aspect of the localisation process is how the images of the vision cameras can be mapped in relation to the machine. The combination of the position of

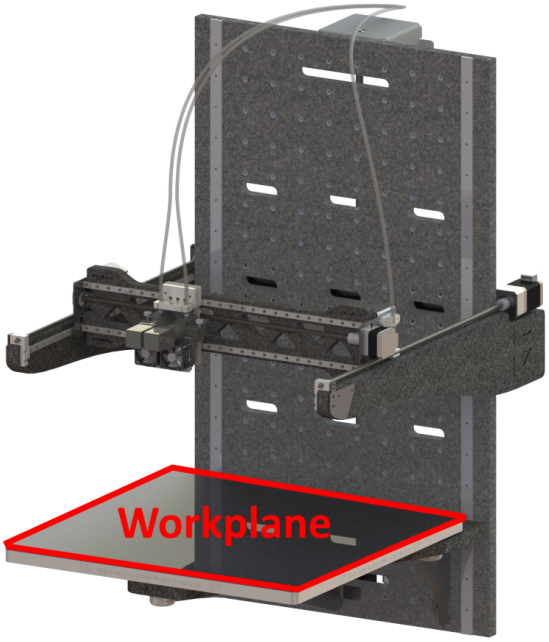


Figure 4.2: The Workplane on a manufacturing machine (in this case a 3D printing module on a standard equiplet frame).

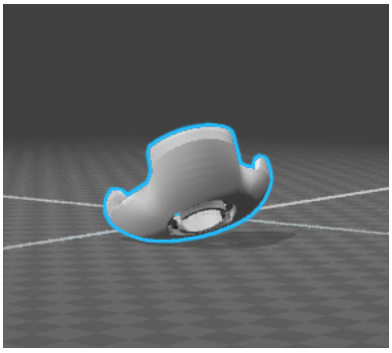


Figure 4.3: Impossible (unstable) angle of an object.

the camera, the possible position of the object, and the object model gives a large amount of information, which can be used for the identification and localisation process.

4.3 Proposal - Generic Method

The metadata and design parameters lead to a proposed generic method, designed for use when both the 3D product model and sufficient metadata (like camera positions) are available. The method can be performed using two or more 2D cameras for 6D localisation of an object. The generic method is shown in Figure 4.4.

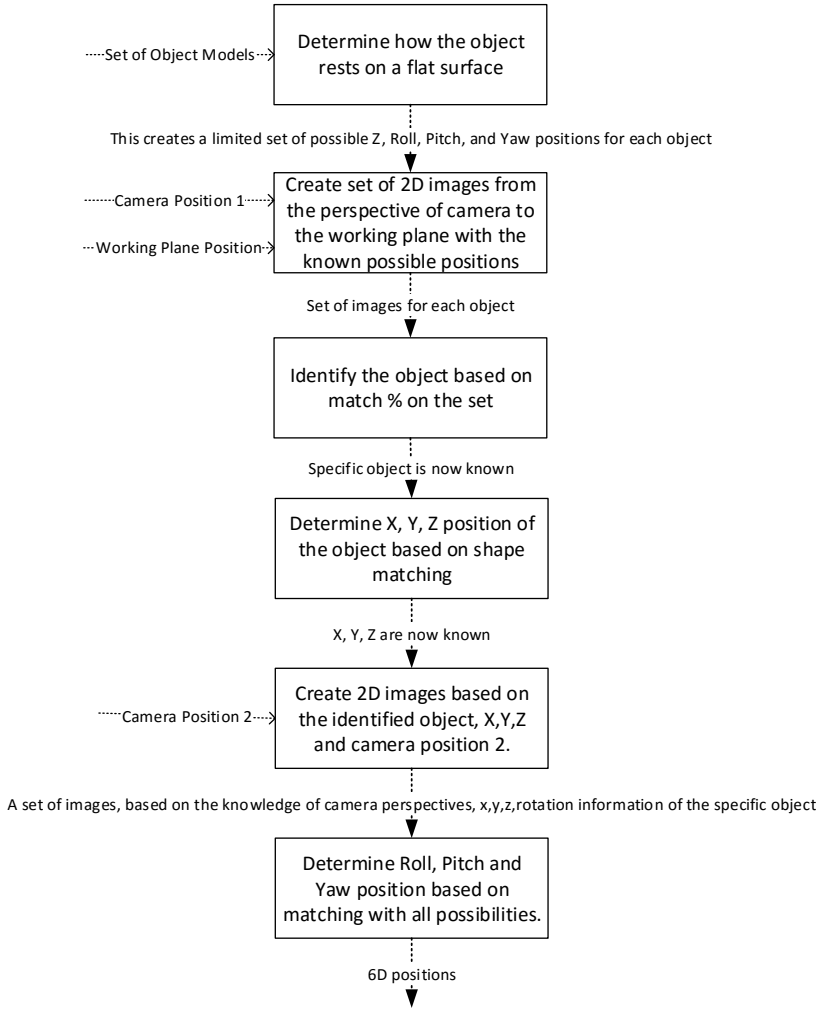


Figure 4.4: The 6D detection using metadata in multiple steps.

The image shows all conceptual steps and the data that is used and generated to come to the recognition and localisation of an object. The first step is where the models of the objects that could be present are given. These models

are analysed to determine how they can be positioned on a flat surface. The analysis creates a number of possibilities that are put into a set, showing the object with the possible roll, pitch and yaw positions. This information is fed to the second system. The second system uses live data that holds the configuration of a live system, including the workplane and camera positions. The data is used to create a set of 2D images that emulates the camera's view by using a simulation that places the model in the possible positions on the workplane from the perspective of the camera. The images are used in the real 'runtime' machine vision process when the object reaches the machine. Object matching with the top camera confirms which object is at the machine and determines the X and Y positions and infers the Z position using the workplane location. However, the matching process needs to be repeated, since one 2D camera cannot precisely determine the roll, pitch, and yaw of the object. Therefore the steps are repeated using another set of simulated 2D images made from the perspective of a camera from another angle. This leads to a precise measurement of all 6 dimensions.

The method could be used for all kinds of object recognition and localisation so long as (1) the object is in rest on a known flat surface, (2) the object model is available, and (3) if there are 2 or more cameras with different perspectives. Precision is based on the camera perspectives, resolution, lighting and model accuracy.

4.4 Design

To create the proposed generic vision method, number of systems need to be designed.

4.4.1 Requirement decomposition

This subsection will give a further decomposition of the Equiplet Functional Requirement 7 (EFR7), as mentioned in Chapter 3, and its two sub-requirements mentioned at the beginning of this chapter. These are split into two different systems, the tool and vision system, which will have the following functionalities:

Tool

- to parse to STL vertices from a CAD model file.
- to define one or more camera positions and orientations of the cameras.
- to simulate the workplane position.

- to render a 3D object based on vertices.
- to generate (2D) images based on the defined camera positions and orientations.
- to perform basic image processing routines on the images for vision purposes, e.g. grey-scaling.
- to save images in a set for later use.

Vision

- to identify objects
- to accurately define the 6D position of an object.

4.5 Implementation

Given these functionalities, a decomposition can be made for the actual implementation. The following four modules were developed:

1. The preprocess - which determines which orientation an object can have.
2. The modelling tool - which generates 2D images from a model based on the camera perspective
3. The vision module - which recognises the real objects by using feature detection
4. The 6D process - which uses the images to determine the 6D positions

This leads to the use of the following modules that were developed, see Figure 4.5.

Several steps in the process can be combined into different sub modules, which can be used in a generic manner. This makes the software more modular and reusable for diverse purposes.

The implementation is based on the original design and specific non-functional requirements, which are mostly technical or practical criteria, but do have an impact on the choices. For the implementation three non-functional requirements are added to the requirements:

- The entire process should be performed within several seconds to be able to be used in an automated manufacturing process.
- The software should be able to run on a multi-platform environment.

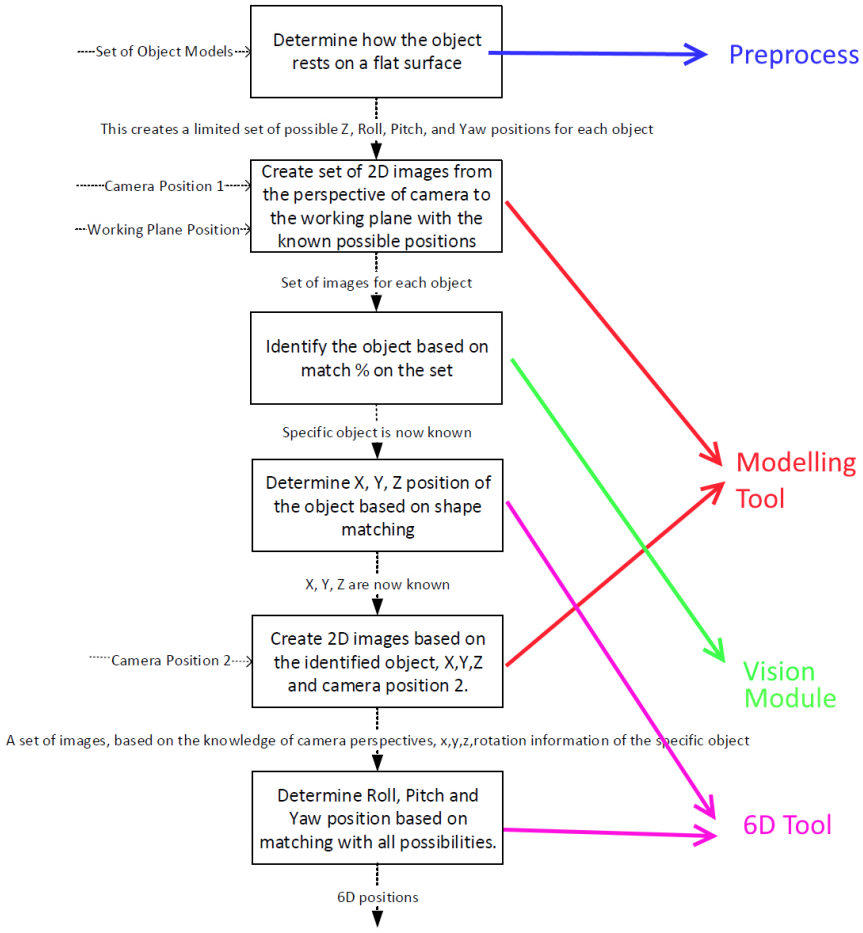


Figure 4.5: System decomposition for the 6D object recognition and localisation process.

- The models should be able to handle the Standard Tessellation Language (STL) model format (which is commonly used by industry)

These technical cases led to the code being developed in C++, using the QT framework. Also the product models use the STL format, which is based on vertices, see Figure 4.6

Since an object is described in vertices it is more efficient to store in a binary format. Also, STL files are often used for prototyping and additive manufacturing purposes. The next step is to define the implementation of the required software modules.

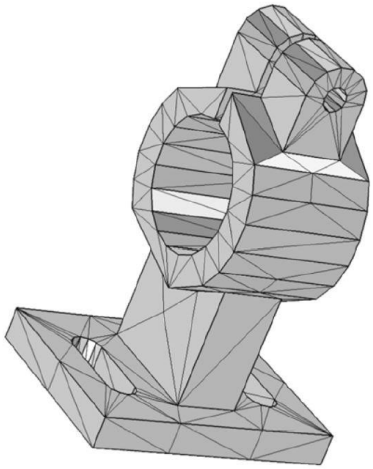


Figure 4.6: An object model based on the STL format is all made up of vertices.

4.5.1 Preprocess

For the first step a pre-process software module should be implemented that analyses the 3D model to determine the possible orientations that can be used to create an image for all possible orientations of the object on a flat surface. This has been designed and is shown in Figure 4.7.

The preprocess uses the model and places it in all possible orientations where it is at rest. The model is visualised in a framebuffer where an image is saved for every possible orientation. During the design phase the possible orientations were already known, since they were given as metadata belonging in the object. However, the process of determining the possible stable orientation on a flat surface could also be automated by analysing the model. In future work this analysis could be achieved by calculating the centre of mass and surfaces to define all possible orientations of an object.

4.5.2 Modelling Tool

The second step is the design of the modelling tool. This tool will be able to generate 2D images from the perspective of the camera for all possible positions of the part lying on a known surface (the workplane). A number of actions can be taken through the GUI, see Figure 4.8 including changing the position of one or more cameras, changing the workplane and adding different STL objects. The tool will then create a view from the camera perspective

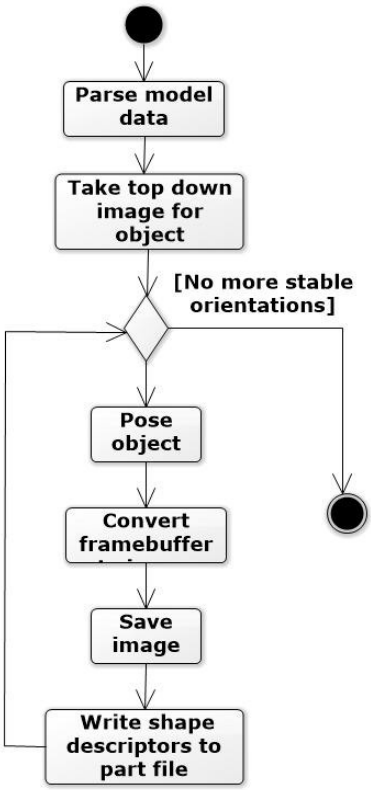


Figure 4.7: Preprocess that creates a dataset for the vision system by using the possible orientations of a model at rest on a flat surface.

and use the OpenGL framework[1] to render it. The tool uses this to grab an image and save it to file. By repeating the process for all possible options that the preprocess defined, an entire set of images is produced that can be used for the vision and 6D localisation process.

While the tool provides ways to manually produce the images it must also be able to run without user input. This makes it possible to automate the process to create images for the 6D localisation process on demand. Hence, Figure 4.9 shows the activity diagram for the tool without the use of a GUI. The tool can be used by the command line and directly called from another process.

[1]https://www.opengl.org/ - last accessed 29-04-2016

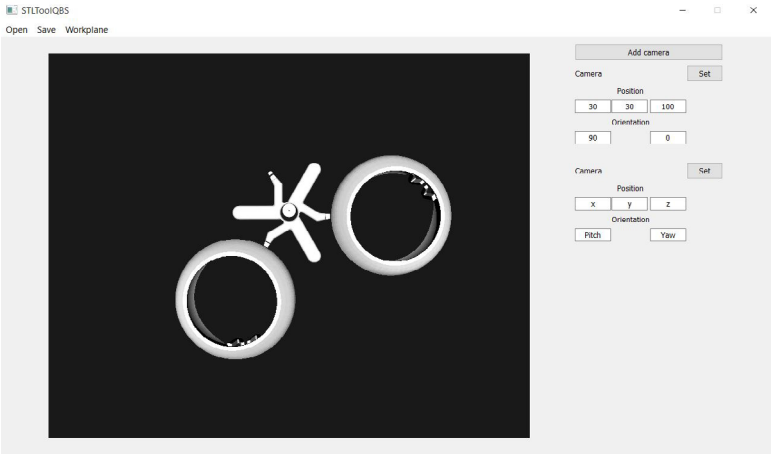


Figure 4.8: The model (STL) tool with GUI, here you see the rendering of an object based on manual positions input.

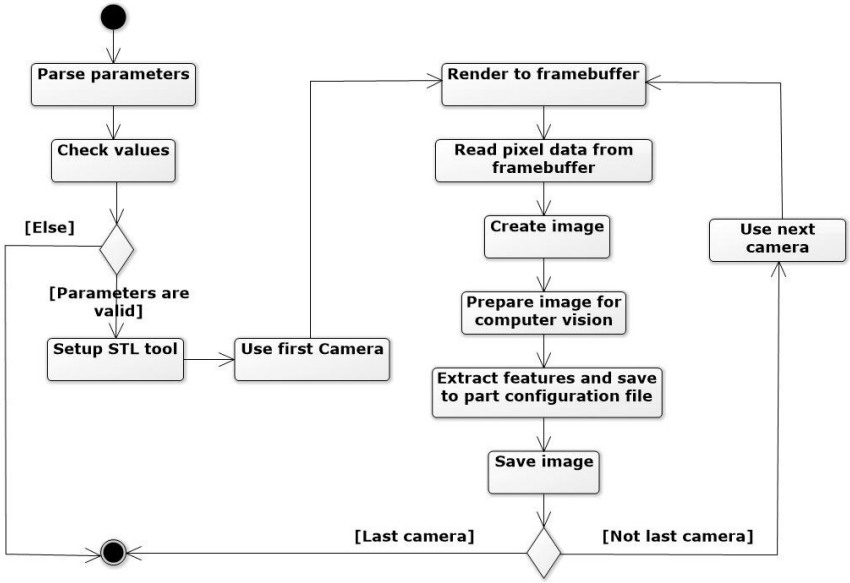


Figure 4.9: Activity diagram for the modelling tool without GUI.

4.5.3 Vision Module

The vision module must be able to perform standard image processing routines and feature detection to recognise the object. Image processing is required to adapt to specific lighting conditions. Including removal of noise, threshold

that adapts the lighting levels and clipping algorithms that can distinguish the difference between objects and the background. These are standard routines that can be used from any number of vision libraries.

Since a limited of set of objects, and a detailed set of images are now available with the correct 'live' perspectives it is now easy to implement the vision modules. Five standard features are implemented:

- Width.
- Height.
- Aspect Ratio - The relation between the Width and Height.
- Number of Holes - Amount of holes that can be identified by identifying connected parts of background within the objects boundaries.
- Area - The total surface area of the object.

These five features can be implemented straight forwardly and used to match the 2D images with the real image. However, in this case these are sufficient, since the image set is limited to the specific possible objects and positions that are given by actual product agents that have been scheduled at that time.

Figure 4.10 shows the steps of the computer vision process. The image taken from the camera is first adapted to the current lighting using thresholding algorithms, then different objects within the image are detected, possibly combined when they are connected, and filtered out. Multiple objects can be within the image and are checked one by one. Each object is analysed on specific features (which can be extended when necessary). Finally the results of the feature detection are stored and compared with the set of possible images. Thereafter the results are matched with the model to establish which object has been detected.

Figure 4.11 shows an example of the matching process with a specific part.

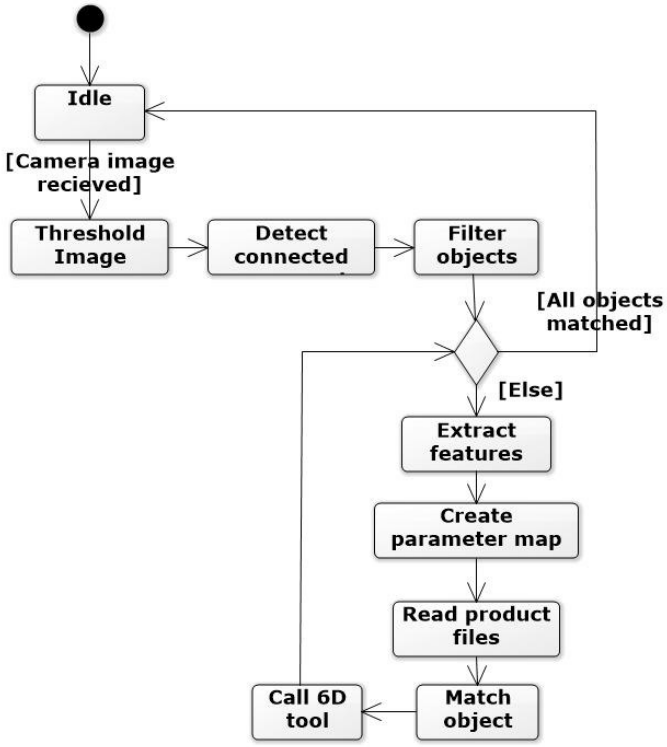


Figure 4.10: Matching the object using feature detection with Computer Vision.

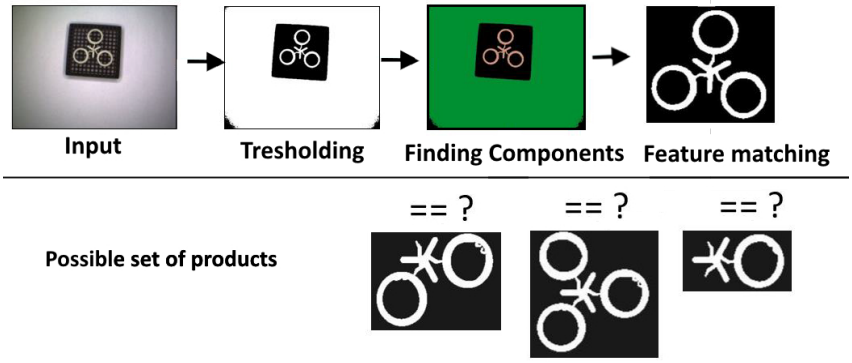


Figure 4.11: Extracting objects.

As shown in the figure, the part is lying on a tray, and the step from input to thresholding shows the result of the thresholding process; effectively taking away the background and exposing the object. In the image, only one object

is found during the finding components process. Then the component is cut out to be used for the feature matching with a set of known objects.

After the feature matching process the 6D process is started that needs to define the locations.

4.5.4 6D Process

Based on the requirements the tool for the 6D process is implemented; the process is shown in Figure 4.12.

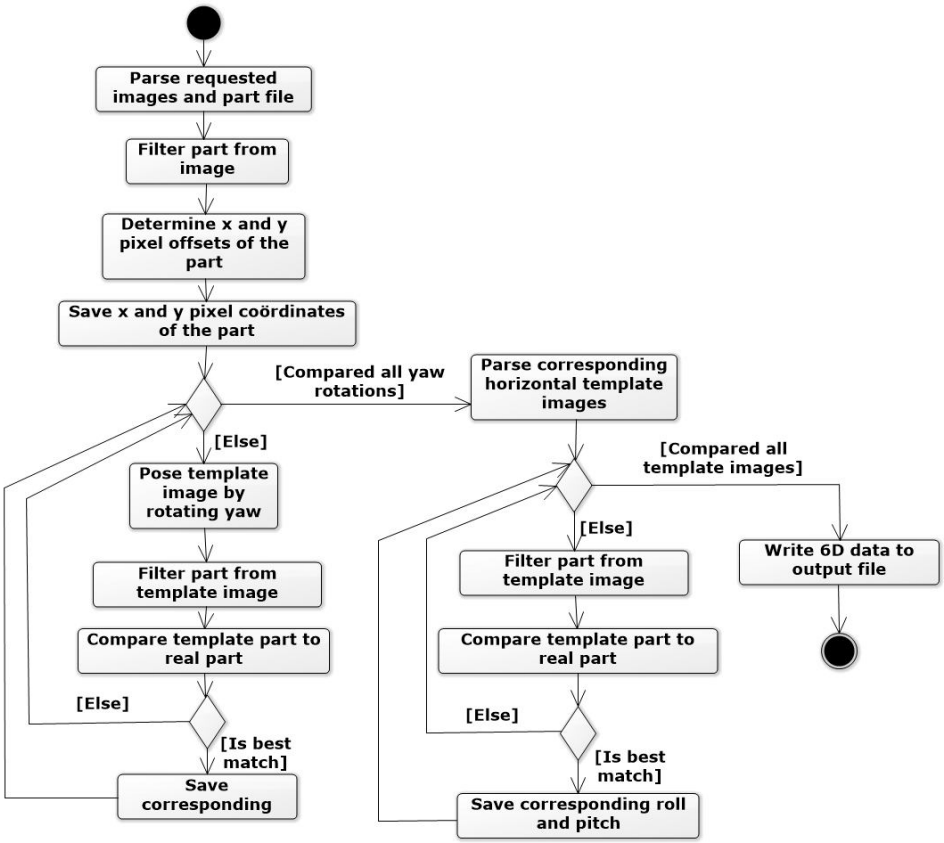


Figure 4.12: 6D localisation process.

The 6D process uses the images and positions of the feature, to determine the positions of the object. At this point the object and all possible orientations are already known. The process starts with filtering the object out of the image, after which the X and Y positions can be determined. The next step is to determine the correct yaw (rotation). The determination is performed by comparing the real image with the test set, which uses a template image.

The process also determines the roll and pitch positions by comparing the templated image set with the real image. When a match is found the steps are repeated with the perspective of the horizontally placed (2nd) camera. If multiple high matches are found, they are checked by comparing the results with the other orientation matches. So, the results can be verified, since the yaw and pitch orientations need to match to have an (unbroken) symmetrical model. After the best match is found, the process determines the complete location of the object in all 6 dimensions.

4.6 Proof of concept: Floe

The process has been tested using a number of product parts. One of them is a specific object called a 'floe'. A floe is a part of an electrical shaver. We take the 'floes' as an example to illustrate the whole process. A floe is an object that is created either using 3D printing or injection moulding. The object is made of 1 to 3 rings connected to a carrier. This object is particularly challenging since at times it can be positioned in an unexpected way. Figure 4.13 shows the floe, both in a correct and incorrect positioning.

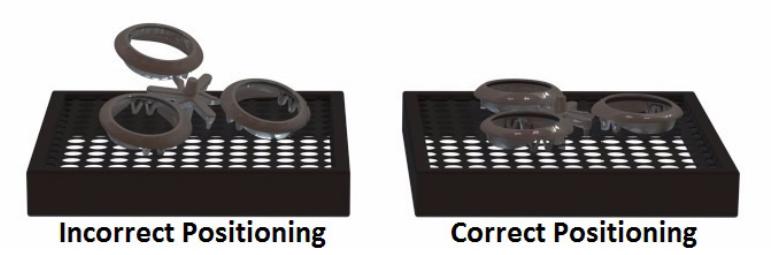


Figure 4.13: The need for 6D localisation when a floe has been positioned in the wrong way.

The problem is that from a top view camera it can be difficult to distinguish the positioning of a floe to see whether it is placed correctly or incorrectly. So while this object is actually on a tray that fixes its position, it still needs to be checked in 6D to be sure that it can be handled. Hence, an equiplet was configured with 2 simple, i.e. low-cost and low-resolution, 2D industrial cameras that view the workplane from different angles. Figure 4.14 shows the test set-up for the test. The equiplet is also configured with a LED ring for lighting, and a workplane.

The test makes use of the 3D tool and 6D process to determine which flow is used and what its position is. The yaw is determined by turning the template image 1 degree at a time until the highest match has been found. After that the roll and pitch are determined, based on possible positions that

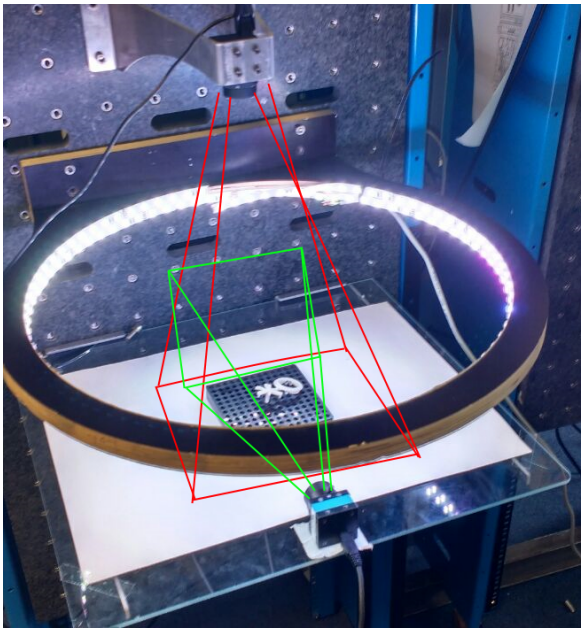


Figure 4.14: An equiplet setup with 2 cameras from different perspectives.

are known by the product agent.

Figure 4.15 shows an example with the floe, with in the left image the real image and the right the precise match of an image with the modelled image.

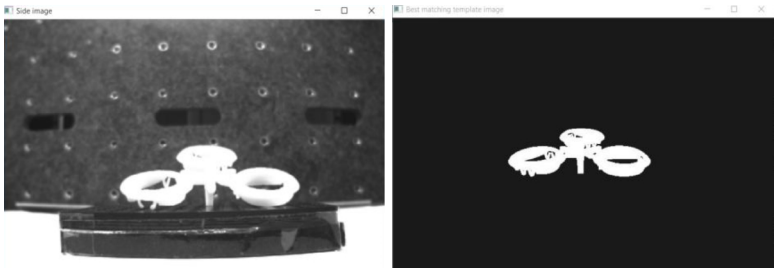


Figure 4.15: The tool generated image matches the real object.

In the test, three types of floes with either 1, 2 or 3 connected rings were put in different positions. Either tilted, i.e. wrongly put in the tray and put in different positions within the workplane of the equiplet test setup. Table 4.1 shows the results.

The test demonstrates that two cameras can effectively determine the 6-Dimensional positions of an object. Match percentages are between 80 and 97% and thus high. In this case, all tests ran correctly and gave precise

Table 4.1: Test results, based on the floe test case. # = Test-number, P = Positioning, Type = Part-type, X, Y, Z = X, Y, Z in pixel coordinates, R, P, Y are Roll, Pitch, Yaw in degrees, RP% = Roll/Pitch image match in %, Y% = Yaw match in %, Time = non-optimised execution time in s.

#	Pos.	Type	X	Ys	Z	R	P	Y	RP%	Y%	Time
1	Flat	Floe1	371	276	229	0	0	324	98	82	1.138
2	Flat	Floe1	339	235	230	15	0	90	83	97	1.133
3	Flat	Floe1	309	268	228	0	0	188	93	97	1.090
1	Tilted	Floe1	326	255	214	15	0	198	94	91	1.136
2	Tilted	Floe1	333	241	210	15	0	88	92	89	1.141
3	Tilted	Floe1	379	238	212	15	0	18	85	87	1.128
1	Flat	Floe2	326	259	225	0	0	332	86	87	1.187
2	Flat	Floe2	312	265	223	0	0	264	83	94	1.200
3	Flat	Floe2	351	253	229	0	0	98	90	97	1.162
1	Tilted	Floe2	353	247	208	11	19	98	94	87	1.190
2	Tilted	Floe2	334	239	208	11	19	172	96	80	1.236
3	Tilted	Floe2	328	269	209	11	19	288	96	85	1.179
1	Flat	Floe3	330	262	230	0	0	322	86	97	0.584
2	Flat	Floe3	343	254	232	0	0	166	81	98	0.596
3	Flat	Floe3	343	254	231	0	0	88	87	96	0.596
1	Tilted	Floe3	343	262	185	11	19	84	91	84	0.597
2	Tilted	Floe3	326	259	180	11	19	208	89	87	0.596
3	Tilted	Floe3	334	279	197	11	19	348	92	82	0.568

matches of the object. Execution time was also quite high, with a maximum around 1.2 seconds. However, currently no optimisations were used and this could likely be optimised in the future.

4.6.1 Accuracy

Currently the most important limitation is that the object needs to be detected correctly. As long as this happens the theoretical accuracy of the 6D localisation is 1 pixel on the Z, X, Y positions, and 1 degree on the Pitch, Yaw, and Roll angles. However, the accuracy is purely based on two variables that could easily be adapted. Mainly the resolution of the cameras, and the size of the test set for the pitch, yaw, and roll angles. As such the x, y, z accuracy can be increased by either using subpixel detection (Rockett, 1999) or increasing the camera resolution. For the current case a set of 180 images per dimension was used for the angles. This means that each image has a 2 degree offset with the next. Hence, when matched correctly the acccuracy should fall within 1 degree. By increasing the test set with a smaller degree offset, the theoretical precision could be increased. Of course, these accuracies are under the assumption that the camera's position is exactly known, and lens correction has been performed accurately.

4.6.2 Limitations

An important limitation is that the CAD models should be accurate, and that objects should not touch each other, as otherwise they will not be seen as separate objects and will not be detected. A risk that can also occur is that shadows from one object cannot be distinguished from the object (or other objects) itself. A similar problem is when not sufficient contrast is available between objects, or between the object and the background. As such, good lighting from all directions is important. However, in common implementations these limitations should be easily mitigated. Designing objects using CAD models has become more mainstream, and using a LED ring or other lighting can be easily set up when using reconfigurable machines.

4.7 Discussion

The current chapter shows a method that is both simple to use and flexible. The process makes use of the modelling (STL) tool and some simple, but effective processes to detect and localise objects in 6D. What makes the process unique is the use of the knowledge of the object and the configuration of the machine itself. This creates a flexible and dynamic approach that can be applied in diverse settings.

Some aspects can still be improved. For example the current implementation is based on the knowledge in the object of how it can rest on a flat surface or tray. While this 'stable position' information might be easily added to a product agent, it might also be straight forward to be determined automatically. This could be achieved by analysing the surfaces and centre of mass of the object.

The current method is very effective, since the set of objects that need to be recognised is always limited, as an equiplet agent will always be able to determine by communicating with the product agents nearby which objects are currently within range. However, if applied in different situations this information might not be available. In such a situation, a larger database needs to be queried. The use of a large object database could result in high match rates for different objects, since there are more small differences and more models that need to be matched against the current objects. The risk of confusion between parts could be mitigated by adding different features that are validated, making the matching process more specific for different objects. However, for the current test cases a limited amount of simple features proved robust and effective enough against the tested objects, even when extending the set with more and different objects.

4.8 Conclusion

Below the two sub research questions RQ1a and RQ2b will be discussed: First, *RQ1a is restated: What data is available and how can the data, that both product and machine contain be effectively combined to help the identification process?* The most important data that needs to be available for the approach is (1) the product model, (2) its possible stable orientations on a flat surface, and (3) the camera positions in relation to the workplane (or any other known flat surface or tray on which the object lies). This data makes it possible to use a straight forward implementation approach and use two 2D cameras to identify and precisely localise the object.

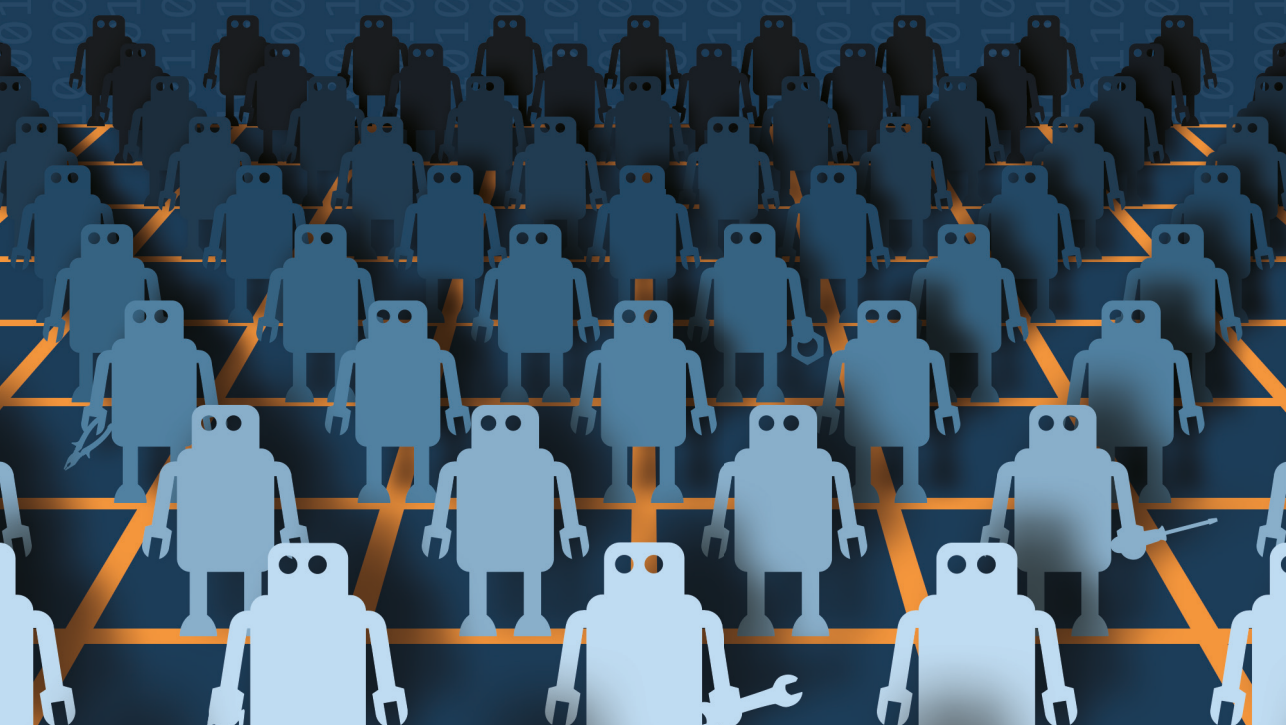
Then to restate *RQ1b: Can the 6D localisation effectively be performed by 2D camera systems?* When the meta data is available and other limitations are taken into account it is effective to use this method. The method is also simple to adapt. The STL tool can adapt the images for different camera positions and models of different objects can be added in real-time. While this chapter only specifically discusses one proof of concept using a typical product part, the process has been experimented with on different configurations and products with similar results. Hence, we are confident that the system is quite flexible in use and can be applied in different situations.

The two answers brings us to the main research question for the current

chapter: *RQ1. How can the detection and localisation of (previously unknown) objects be simplified and generalised?* This can be done by (1) making use of the knowledge of different entities within the system, (2) combining that information and (3) creating a tool that it can use to dynamically compose the required images to match these against the real world. Here you can see the Cyber-Physical aspect again, where the virtual 'Cyber' world and the real physical world work closely together to create a straight forward solution to a practical problem that is commonly quite difficult to solve.

In general, the conclusion is that the generic method for 6D localisation is effective, as shown in the results the vision system was able to identify the correct object and provide sufficient precision (within 1 degree or pixel) for the equiplet to handle the part. It is a good example of Cyber-Physical Systems working together and simplifying a problem such an object detection and localisation. The use of the models and the communication between the equiplet and product make this approach possible, increasing flexibility and solving the problem of object awareness in a changing environment.

05



Reconfiguration

“I suppose it is tempting, if the only tool you have is a hammer, to treat everything as if it were a nail.”

– Abraham H. Maslow

Reconfiguration

Chapter 3 proposed the concept for Grid Manufacturing. The concept of Grid Manufacturing is based on the idea of autonomous entities that cooperate for maximum flexibility. However, it does not yet describe the means of how this was achieved. Hence, the current chapter will describe one of the most important aspects of the study: reconfiguration. Reconfiguration in this sense is not only the ability to change the 'configuration', i.e. changing the hardware modules, but more specifically the ways it adapts its software and the ability to be able to use it. The 'agility' that is created with these aspects has to be made by software.

Equiplets are reconfigured by changing the hardware modules. From the viewpoint of autonomic computing, this falls within the category of self-configuration (Computing, 2003). The equiplet can use its reconfiguration behaviour to change its services. Modules, like grippers, parallel manipulators, and camera systems can be changed to offer new capabilities to the machine and therewith new services to the products. However, changing a module has a number of effects:

- It affects how it can interact with a product, i.e. the product knows which services it needs, but the product is not aware of how to control hardware directly and therefore cannot control the hardware itself.
- It affects how the equiplet is controlled, i.e. each module has its unique interfaces and micro-systems that need to be controlled.
- It changes the service and capability of an equiplet, i.e. what it can do and with which limitations.

These problems have to be resolved to be able to quickly adapt the manufacturing systems to new product demand. If the manufacturing system works autonomously and offers a generic service they would be able to be changed on demand at any time without interfering with the other manufacturing systems. This would create flexibility to add, remove, and change services & abilities at any time. Simplifying the reconfiguration process could also lead to three opportunities:

1. When there is sufficient redundancy in the grid, systems with a low utilisation could be reconfigured to provide a service that is currently high in demand to improve overall efficiency.

2. A system could temporarily be reconfigured with new hardware to experiment with new services that a (prototype) product requires, i.e. real 'live' production grids could be used for automated product prototyping, which will likely be useful to find problems in the manufacturing process early in the product development phase.

3. Updating or repairing hardware can be performed with limited or no impact to the grid in general.

To be able to utilise these opportunities the following problems have to be solved.

5.1 Problem Description

One of the main requirements for reconfiguration is that it should be simple, fast, and not limit the flexibility. Hence, it should be possible to reconfigure a system without the need to program and build new software to let it perform its service. Reprogramming would limit the ability to change the system on demand and as such would inhibit the ability of the system to automatically manufacture new custom products. Reconfiguration introduces a related problem: a product is not aware of which manufacturing system will produce it. Scheduling takes place dynamically, based on the demand and supply of the services that the product requires and which the manufacturing systems supply. Services are designed to be standardised and as generic as possible in such a way that as many as possible products can use them. As a result, both product and the manufacturing system are not designed specifically for each other. Hence, to be able to use the service for a product they should be able to interface and understand each other. This asks for an ontology that both product and manufacturing system can use. The architecture should take into account which services and limits it can provide and match these to the requirements of the product.

5.2 Additional Research Questions

The main question for the chapter is: *RQ2 Can reconfigurable manufacturing systems be controlled without the need to reprogram them for every new product or hardware module?* The main question for this chapter is split into three additional sub-questions:

RQ2a How can the product use the machines when it has no knowledge of the manufacturing system or its (hardware) configuration?

RQ2b How does the machine know which services it can provide after it has been reconfigured?

RQ2c How can the machine control its (changing) hardware modules?

5.3 Literature Overview

Van Moergestel first introduced the idea of Product Agents that use 'Product Steps' to describe the necessary actions that a product has to take to be manufactured (Moergestel et al., 2013b). However, in his research the product steps were described in detail and as such the manufacturing systems could always directly complete them. In other words, the product was created by a limited and standardised set of steps that were well-defined and known by both product and machines. The manufacturing machines were only capable of directly performing these steps. Hence, the system is limited by these steps and should be reprogrammed if new steps are added or the configuration is changed. Therefore, another system is required that gives more flexibility. The goal is to create a way that products can immediately use new configurations or services that become available without the need for taking the manufacturing system offline for a longer period of time to change its programming.

Basically what is required for reconfiguration without reprogramming to work is for different unknown systems to interact. Interaction or handling of objects is also a matter of research for game and simulation systems, which often speak of *smart objects*. Kallmann and Thalmann propose a framework that handles many possible interactions between virtual humans and the environment (Kallmann and Thalmann, 1999). However, while in this approach humans do learn to utilise the objects by reasoning, it only searches for suitable behaviours that fit the smart object.

Gibson introduced the concept of *Affordances*, which shows a relation between an object and an organism, i.e. it shows the possibility of an action (Gibson, 1977). An example of Afforances is the handle of a bag, which clearly provides the affordance to carry it above the ground. While Gibson applied this to organisms in natural environments the notion of affordances has been spread in various fields, including robotics. From the theory of affordances, actions that can be performed should not be perceived as properties of a system, but more as a possibility of which an actor becomes aware.

From a more specific technical perspective there are several studies of interest. The literature provides a number of distributed production systems that use agent technology. However, most of these are used in the context of distribution of resources (Barata et al., 2008; Khalgui and Mosbahi, 2010; Cho and Prabhu, 2007; Trentesaux, 2009; Wang et al., 2009). Barata for example uses

a true reconfigurable system. However, this system is still controlled by a central hierarchical control system and is therefore not as flexible or distributed in the meaning that all systems work autonomously (Barata et al., 2008). Wang similarly uses distributed real-time information to create a centralised cyber workspace that is still centrally controlled (Wang et al., 2009).

Trentesaux looked into distributed control systems and even distinguishes three classes: Class 1: Centralised, Class 2: Semi-hierarchical, Class 3: heterarchical (all systems are equal). Based on his classification the grid manufacturing concept will be a true Class 3 system, where all products and manufacturing systems will be equal and cooperative in the heterarchical sense.

Based on the related work it is presumed that there is still need for a system that is both truly distributed, and in that sense, heterarchical i.e all systems are equal and autonomous, and reconfigurable. Hence, we introduce a system that falls more in line with Grid Manufacturing by introducing a new concept.

5.4 The Concept of Product Step Translations

The objectives for reconfiguration are based on the specifications as mentioned in Chapter 3 Concept. The most important requirement is that no system needs to be reprogrammed when it is reconfigured or a new product is designed. Also, products are not specifically designed for the manufacturing hardware. Hence, there needs to be a translation from the basic product step that the product uses to describe its manufacturing process, towards the instructions that the hardware will need to be able to perform those steps. In classic manufacturing, all parts and manufacturing steps had to be explicitly defined to be able to automatically produce a product. Customisation and replacement of parts were handled before the manufacturing phase. However, with the approach of grid manufacturing, a product can be manufactured dynamically. Hence, there is the possibility to use product customisation in any phase of the manufacturing process. This opens the door to product abstraction, which is seen as an explicit abstraction in the product manufacturing design. With product abstraction, parts of the manufacturing process can be interpreted by the grid on demand. For example, different manufacturing hardware might be capable of performing a similar service. In some cases product abstraction can even be taken further, e.g. from the design perspective, if a colour has not been chosen for the product, the grid itself can choose to produce the product with any colour based on parameters, like cost or availability of specific equipment.

Based on the concept of product (manufacturing) abstraction, a vision emerges where the manufacturing process begins from a high abstraction and is

translated step by step to specific instructions that can be sent to the hardware. This concept can be seen from two different perspectives: from a design point of view, i.e. the customer does not explicitly define the entire product; and from a manufacturing system perspective, i.e. product steps can be produced by a number of different (hardware) systems. In both cases the system that will perform a product step is not known *a priori* at the moment that the product manufacturing process begins. The layered model, as seen in Figure 5.1 is used to show which steps are taken to delineate.

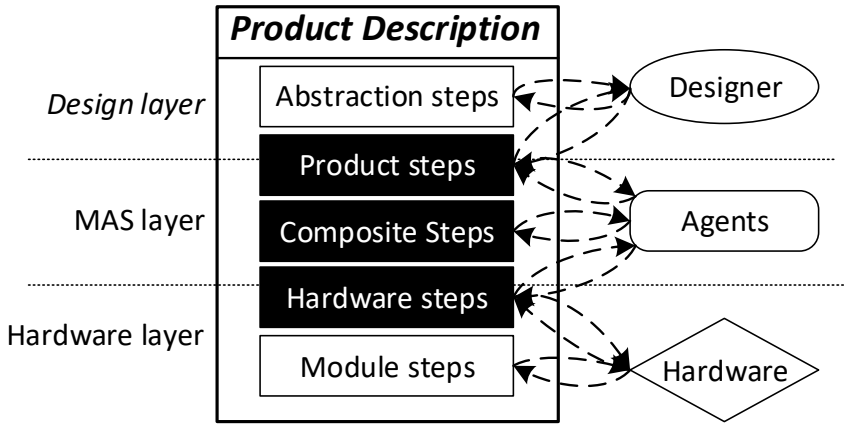


Figure 5.1: Product description from an abstract design to specific instructions performed by the hardware.

The product description is based on five levels that each add more specific information to the required manufacturing process. In this model the top layer is the most abstract and is based on the original wishes of the customer, while the bottom layer consist of the low-level instructions that control each module in the equiplet performing the actual manufacturing steps. In this model the manufacturing process is described by steps, which will need to be translated to the next level by different systems to come to an explicit instruction that will be used to control the hardware. The five steps are defined as follows:

1. The Abstraction Steps are defined by the customer and could explicitly give some options or "dont care" conditions of the design that can later be filled in by the grid;

2. The Product Steps are manufacturing steps that describe (without knowledge of the manufacturing systems) how the product should be manufactured;

3. The Composite Steps define the actions that a grid can perform, which it offers as a service to the products. These services are still abstracted from the hardware that performs them. Composite steps describe the goal of the step that needs to be performed, e.g. place item X.

4. The Hardware Steps are the result of the translation from the composite steps to the specific hardware instructions that are required for the hardware configuration of that equiplet. Hence, a hardware step is no longer an abstraction, but specific for the hardware that will perform the step.

5. The Module Steps adds some runtime information to the hardware steps that were unknown at the hardware step layer, e.g. an object location is given from the vision module towards the pick and place module. In the final step all instructions become atomic and can be directly used to control all hardware modules.

From a practical perspective the model is divided into three parts, the design layer, the Multi-Agent System (MAS) layer and hardware layer. Depending on how explicit the customer wants to define the product, the design can be made either within the abstraction or directly from the product step level. The product steps layer can be automatically translated to service steps, based on the capabilities of the grid. Once a composite step is scheduled to a specific equiplet they can automatically be translated to a hardware step. These last two translations are handled by agents within the MAS. However, the equiplet and modules also have a representation that is used to describe the instructions that will directly control the hardware. This is performed by the hardware layer.

5.4.1 Capabilities

Each equiplet in the production environment has certain properties and behaviours that can be classified as functional capabilities (Järvenpää et al., 2012). In a grid each equiplet has certain capabilities that are provided as service to the grid. All the services in a grid can be used by the product agent to manufacture a product. A capability is defined as a service description combined with the limitations of the service. Examples of capabilities are pick and place, or draw line. Limitations of these capabilities include the boundaries of the workspace. Basically in this study these are defined as follows:
Service = Abstraction of the provided service
Limitation = Min. Max. Weight, Dimension Limits, etc
Capability = [Service, Limitation]
ProductStep = <Service, Criteria>

Criteria = Weight, Dimensions, etc

According to these definitions, a capability is defined as the possibility to actually perform a service step on an equiplet, i.e. all the criteria of the ProductStep fall within the limitations of the hardware configuration.

A product agent divides the manufacturing of its product into product steps. For each product step the agent uses a service of an equiplet. The product agent defines a product step by the required service and criteria of the service. Criteria of a product step include the dimensions and weight of the part on which the service needs to be executed, e.g. one of the criteria would be the dimensions of an object that needs to be pick and placed, so an equiplet agent knows if it is able to perform the service.

5.5 Simplified overview of the Manufacturing Process

Grid manufacturing utilises a heterarchical architecture, where products can negotiate with equiplets in the grid to perform the next necessary step in the production process. The required steps are known to the product agent, which will negotiate the step with a selected equiplet. Since equiplets are modular and reconfigurable they can have a unique configuration. As mentioned before, products are designed without knowledge of the manufacturing hardware. To simplify the translation process, a common standardised set of services is created.

The manufacturing process is described through several actions, which can be processed in parallel by multiple equiplets for several products, making it a continuous process that is performed on demand. The following actions are first explained from an abstract perspective to give an overview of the processes involved. Later in the Chapter they will be discussed in more detail. For every product step the following (simplified actions) take place:

1. Capability checking - Determines if the equiplet is able to perform the product step.

2. Step translations - Translated the abstract product step to specific hardware instructions.

3. Scheduling - Schedules the product step at the equiplet that is able to perform the step within parameters.

4. Step execution - The actual execution of the product step at the equiplet.

The processes will be discussed in the following subsections. However, the implementation and more specific technical details will be discussed later in the implementation section.

5.5.1 Capability Checking

Capability checking is determining if the equiplet can perform the product step. Figure 5.2 shows the steps of this process. Since reprogramming should not be involved according to the specifications it is necessary to make the process data-driven. Hence, the concept involves a database that should be queried to determine which services and hardware configuration are necessary and if the active configuration of the equiplet falls within the parameters.

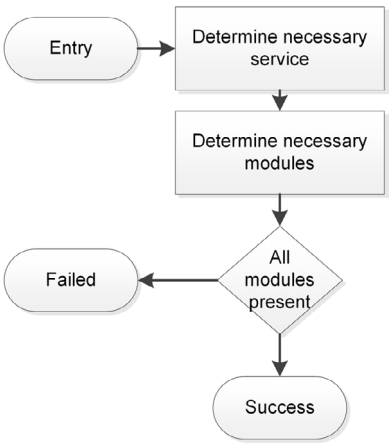


Figure 5.2: The first (simplied) action that takes place in the manufacturing process, the capability check.

5.5.2 Step Translations

The translation step itself needs to be performed in two steps, see Figure 5.3. First, from product step to composite steps, it omits the part information, i.e. it uses theoretical types of parts, without actually confirming which instance of a part is available in the grid at that time (this will be handled during scheduling). The next step is to translate the composite steps to Hardware Steps. Hardware Steps are explicit instructions that are tailored for the specific configuration of modules within that equiplet. After the translations have been performed, the average duration is also known, since the instructions have a known average duration for execution with that specific hardware configuration. Note that the translation process is always successful, since the

capability checking process has already determined if the equiplet is able to perform the product step.

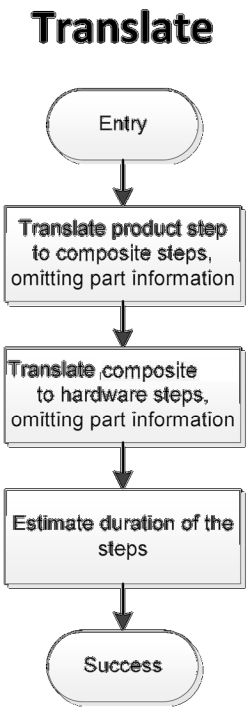


Figure 5.3: The second simplified action, translation actions.

The translation process is basically a delineation process, see figure 5.4. A product step, e.g. place part A on component Y, has to be delineated in multiple actions that need to be performed by multiple modules. The example could require the use of cameras to identify the components, move instructions and the opening and closing of a gripper. As such it involves a number of steps that are completely dependent on the specific hardware. More specific examples and how the process works will be discussed later in the current chapter.

5.5.3 Scheduling

Scheduling in general is considered to be out of scope, since it is the responsibility of the product agent, which is part of the research of Moergestel (2014). However, it is discussed here on an abstract level since (1) it is part of the proof of concept and (2) the translation process is needed to determine the amount of time that the product step takes to perform on the specific equiplet

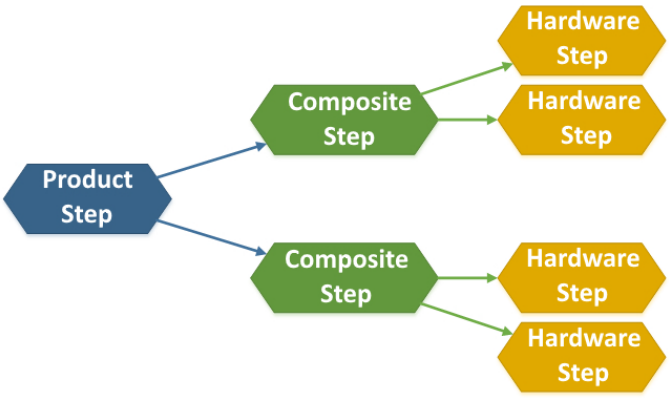


Figure 5.4: Delineation of the different steps, from standardised "abstract" Product Steps to specific Hardware Instructions.

hardware configuration, which is a result of the generic services and the different configurations that might be able to perform the product step. As such, the equiplet configuration directly influences the schedule in a way that can never be determined a priori by the product.

The product agent wants to schedule its next step, see Figure 5.5 for the simplified scheduling process. The shown process is for one step at one equiplet and when failed can be repeated at another equiplet. However, as mentioned before, depending on specific parameters and scheduling, it is not certain that every equiplet can perform the step. This was determined directly in contact with the equiplet. If an equiplet is found, it will determine if the equiplet is able to perform the necessary step. The product agent will communicate with an equiplet to ask if the equiplet is able to carry out the step. If it is capable, the steps will be translated, which will determine how much time it takes to execute the step for this specific equiplet configuration. When the time is known that it takes to potentially execute the process step it is time to find free time slots in the equiplet schedule. If this fails, the product agent has to try again at another equiplet (or request another timeframe). If a time slot is found, logistics will also be required to calculate if the product and its parts can arrive on time for the requested time slot. If the schedule seems possible, the translation information for that specific equiplet is stored and the scheduling phase for this step is a success.

5.5.4 Step Execution

Figure 5.6 gives a simplified overview of the entire execution process, placing the other actions into context. When the scheduled time for a product approaches, the equiplet will contact the product agent and request to start

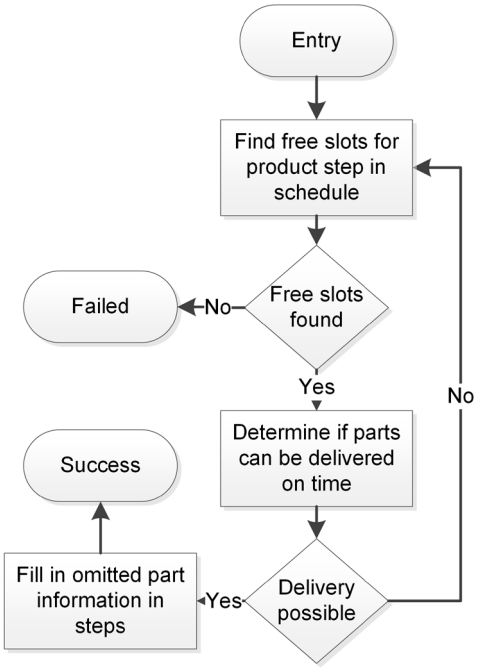


Figure 5.5: Simplified scheduling action of one product step.

the execution of the next product step. The contact will normally trigger a response of the product agent, synchronising the product and equiplet agent. If positive, the equiplet will start performing the next manufacturing step by sending the hardware steps to the different modules in its configuration.

5.5.5 Product Steps

Product steps define all steps of the manufacturing process of a product on an abstract level. In grid manufacturing these steps are always independent of the hardware. Hence, product steps are commonly defined by interactions between parts, e.g a pick and place action where one part is placed on another, or a typical Computer Numerical Control (CNC) action, e.g, a milling machine, where one part (a tool in this case) is moved in a specific path over another part.

Product steps are defined using the type, input parts, output parts, and a set of parameters specific to the type. Depending on the value of these parameters, a single type of product step may result in very different actions.

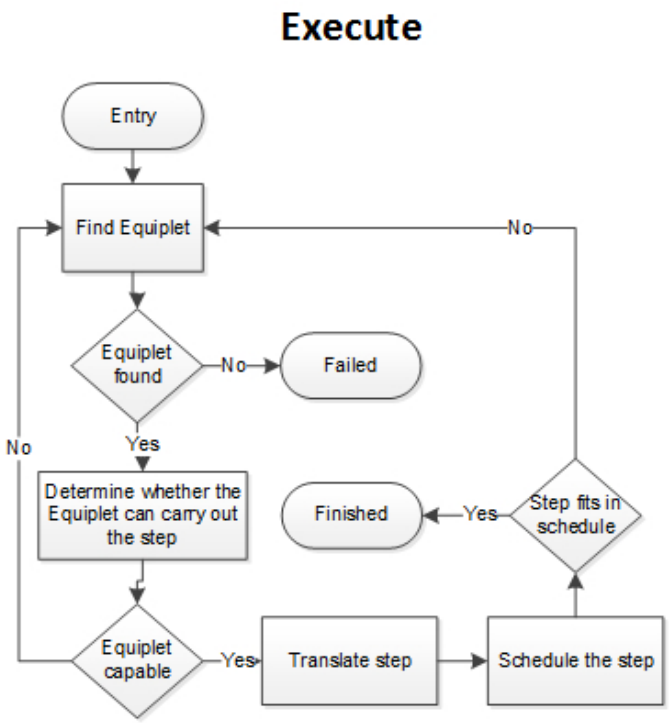


Figure 5.6: Start execution.

An example of a product step type with parameters, using Unified Modelling Language (UML) operation signature, is: Place(subject : part, destination : location). In this case the destination parameter can be relative to objects in the work environment (such as parts). An instance of this product step might for example look as follows: Place(Red Ball, (1,1,1) relative to Crate). This example would result in a red ball being placed into a crate, at position 1,1,1 relative to that crate. In this instance, Red Ball and Crate refer to the part type, not to specific instances of these parts.

5.5.6 Composite Steps

Composite steps consist of more specific actions, and take into account the specific part instance, and where in the work environment they will be delivered. They do not take into account the specific hardware that is used. Each Composite step also has a step type and a set of parameters specific to this type. Every product step is translated into one or more composite steps, e.g., the product step 'Place' shown in the previous subsection would be translated to a composite step of type 'Pickup' and a composite step of type 'Drop'. Both

with the parameters destination : location and safe movement plane : location (the safe movement plane is the horizontal plane in the work environment in which the pick and place robot can move safely). The translation example results in the following steps:

1. Pickup((2,1,1.5) relative to crate A, (6) relative to work surface)
2. Drop((1,1,1.5) relative to crate B, (6) relative to work surface)

Crate A and crate B refer to specific crate entities, as composite steps always refer to specific entities.

A composite step can consist of one or more *mutations*, a mutation is a standardised abstract action that can be performed and describes the abstracted actions that need to be performed for the composite step, e.g. move to location Y and then pick up item X. Mutations are part of the internal handling of composite steps. The Product Steps describes the action from the Product view, e.g. place Item X on location Z. The Composite Steps then translate these to the goals from the machine's perspective, to place an item it first needs to pick it up, by moving to the right point and pick it up (composite step 1) and second move to the 2nd location where the item can be placed (composite step 2). Where the composite step is still an abstraction that needs to be performed by the machine as a whole, the mutations are unique for a separate module, e.g. to pick an item up a gripper is required and an action that moves the gripper. Hence, a composite step 'move and pick up' is decomposed to several 'mutations' that can be handled by different modules.

5.5.7 Hardware Steps

Hardware steps are the result of the last translation. They are explicit instructions for the configured hardware of the scheduled equiplet. These instructions will be used directly by the ROS nodes to control the hardware. Like product and composite steps, a hardware step also has a step type and a set of parameters. Every composite step is translated into one or more hardware steps, e.g. the service step 'Pickup' mentioned earlier is translated into the following Hardware steps:

1. Delta Robot(Move, (*, *, 8) relative to origin)
2. Delta Robot(Move, (2, 1, *) relative to crate A)
3. Delta Robot(Move, (*, *, 3.5) relative to crate A)
4. Gripper(Activate)

5. Delta Robot(Move, (*, *, 8) relative to work surface)

The format being used here is: Module(action : Action, params : Parameter[]). Each hardware step is an instruction for a single module (or multiple modules indirectly, through a single leading module). In this example some parameters, such as coordinates, whose values are not important or can be reused, are indicated by an asterisk(*). These are filled in at the module level. In the case above, the x- and y-coordinates will stay the same, e.g. a deltarobot at 3,4,5, which receives the instruction to go to *,*,8, will move to position 3,4,8.

Figure 5.7 shows an overview of an example where the product step pick and place is translated to specific hardware steps that can be used by an equiplet.

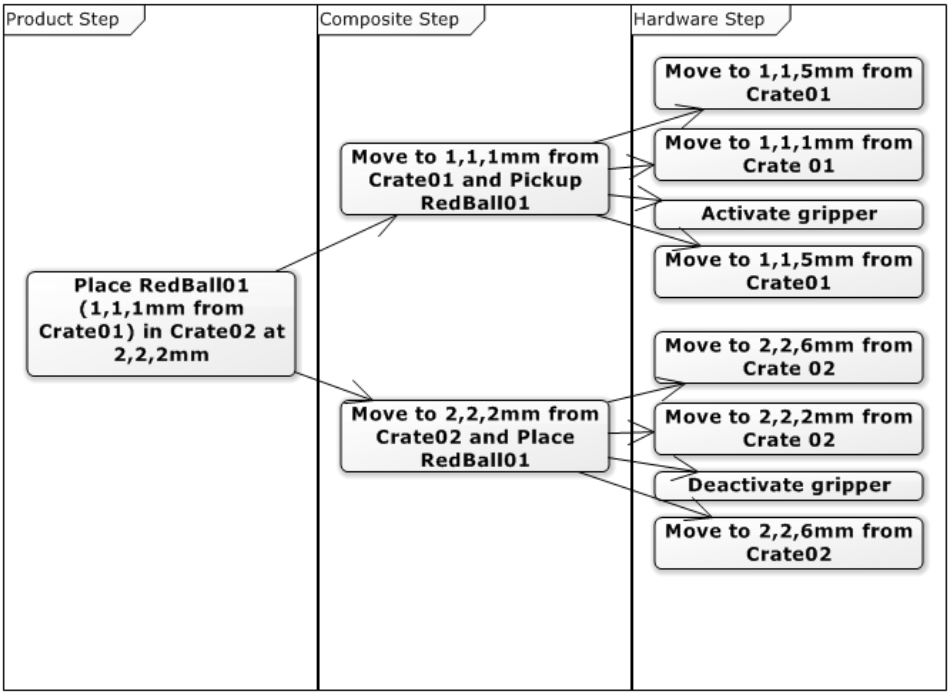


Figure 5.7: Example of a simple pick and place action, from abstract product steps to explicit hardware instructions.

The use of the translation system simplifies the reconfiguration process and increases the flexibility of the entire manufacturing process. This is because products use the abstract product step, which can be translated to the current supply of manufacturing systems as a dynamic process, which can be changed at any time.

5.6 Offering Services

Another part of the reconfiguration process is the ability to change the service or capability that an equiplet can provide. As mentioned in section 5.4.1 we define the capability as the ability to provide the right service and meeting the right criteria to be able to perform the product step.

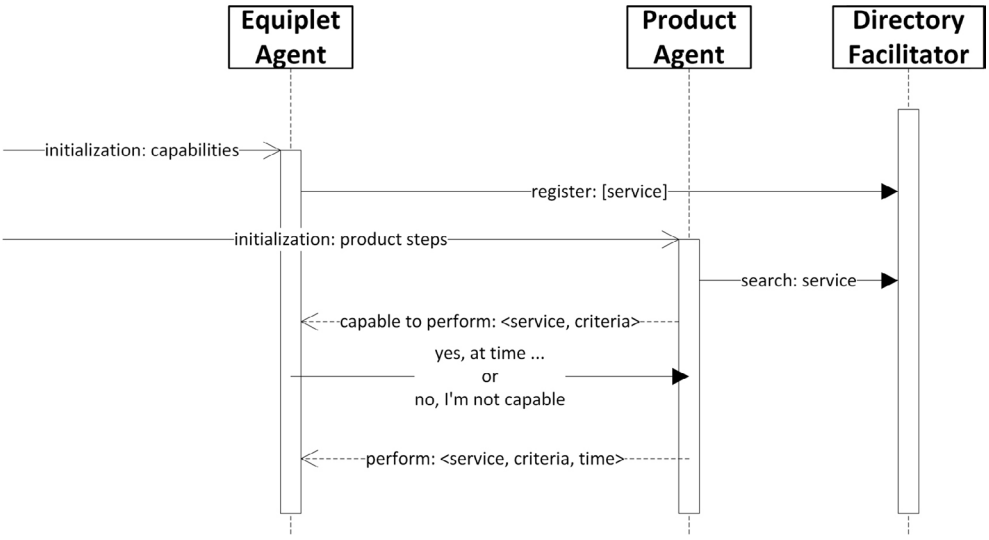


Figure 5.8: After an equiplet starts up or has been reconfigured it sends its services to the directory facilitator.

5.7 Module Configuration

Every equiplet is configured using a number of modules. The modules have a hierarchical reference to each other, which can be depicted in a tree. The hierarchy is required due to the fact that modules can be physically attached to each other. Figure 5.9 shows a Physical Module Tree with a pick and place configuration using 4 different modules. The Physical Module Tree shows the physical relation between the modules, e.g. the gripper in this configuration is attached to a parallel manipulator (or deltarobot) that moves the gripper to a specific position. All other modules are directly attached to the equiplet.

Every module supports one or more 'mutations'. A mutation is a standardised method that describes the action that the module can take. A Composite step can consist of one or more mutations. The mutations that an equiplet can perform can be shown in a Functional Module Tree. The FMT is generated specifically for each configuration. Figure 5.10 shows the FMT for the same

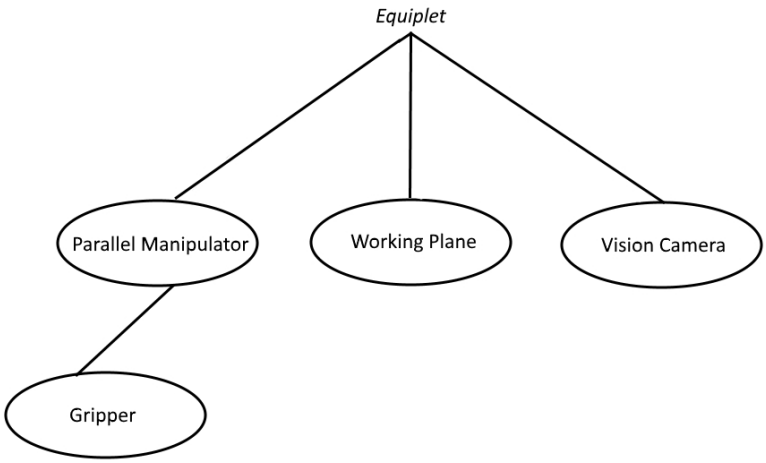


Figure 5.9: Example of a Physical Module Tree for an equiplet with a pick and place configuration.

pick and place configuration as was used in Figure 5.9. Here it is seen that the Parallel Manipulator module grants the 'Move' mutation, and the Workplane module, the 'Locate' mutation. The gripper provides two mutations: 'Pick' and 'Place'. Summarised:

- Each module has standardised methods it can provide, i.e. the mutation;
- To provide a capability certain requirements have to be met, e.g. a combination of modules that can perform the right mutations are required to perform the requested service;
- Each mutation also has limitations to what it can offer, e.g. the weight it can lift;
- The combination of a configuration of modules that match the service requirements, and if the product falls within the limitations set by all modules provides the 'capability'.

A potential risk for this method is the possible influence on the limitations that modules can have on each other. These dependencies are currently mitigated by choosing the limitations wisely. This is performed by hand when modules and limitations are developed. For example: if a gripper has a limitation that it can hold an object with a maximum weight of X, X might be influenced by the speed with which the gripper is moved by a moving module that it is attached to. The potential risk can be mitigated by adding a limitation of maximum acceleration to the gripper. The maximum acceleration

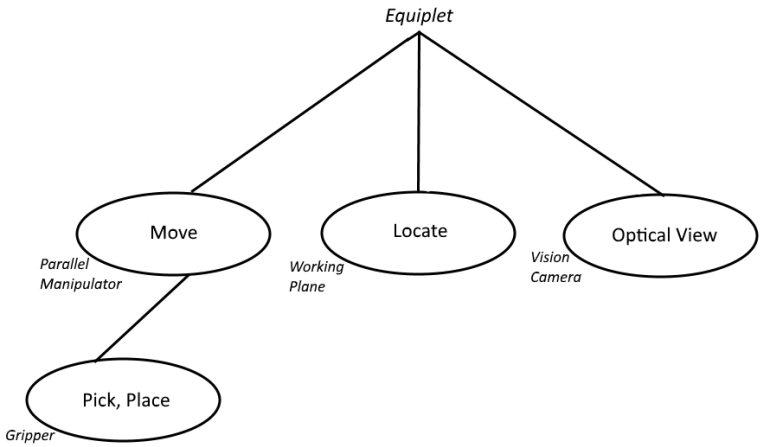


Figure 5.10: Example of a Functional Module Tree for an equiplet with a pick and place configuration.

limit decouples the weight limitation by guaranteeing the gripper will be able to carry the weight so long as the gripper does not increase its momentum by the maximum acceleration.

5.8 Software Design

The abstraction of hardware should be explicitly designed to enable all the mentioned reconfigurable systems. For this purpose a Hardware Abstraction Layer (HAL) has been developed. How the HAL fits into the overall architecture will be discussed in the next Chapter. The HAL has four main functions:

1. To translate the product steps to hardware instructions.

2. To Interface between the higher (logistic) and lower (hardware interfacing) levels.

3. To enable reconfiguration processes of the equiplet modules.

4. To Determine the capability of an equiplet.

Based on these functions and the research on reconfiguration, the design will be discussed in the next subsection.

5.8.1 HAL Overview

To accommodate the main functions of the HAL, the following four main entities have been designed:

- HardwareAbstractionLayer - Is responsible for creating the factories and implementing abstract method calls. Acts on state and mode changes of modules. The HardwareAbstractionLayer class is responsible for the translation and execution processes.
- ModuleFactory - Is responsible for generating Module instances from information stored in the Knowledge Database.
- CapabilityFactory - Is responsible for generating Capability instances from information stored in the Knowledge Database.
- BlackboardHandler - The Blackboardhandler is used as an interface with a Blackboard that can be used to communicate with the low-level systems. Other interfaces might become available as well based on further research.

These main classes use the following seven support entities:

- ReconfigHandler - Is responsible for implementation of the Reconfiguration API (insert/update/deletion of modules).
- Capability - Actual abstract class of a capability. Is responsible for translating ProductSteps into multiple HardwareSteps.
- Module - Class of a hardware module, without further functionalities to translate or execute. Provides specific information of a module (like what it is connected to and where it is mounted on an equiplet).
- ModuleActor - Actual abstract class of a hardware module. Contains the translate and execute methods. Is responsible to make dynamic loadable modules like the Gripper module.
- KDBClient - Implements database calls.
- Translation process - A thread managing the translation of a product step.
- Execution process - A thread managing the execution of hardware steps.

Figure 5.11 shows the (simplified) Class Diagram showing all main classes of the HAL.

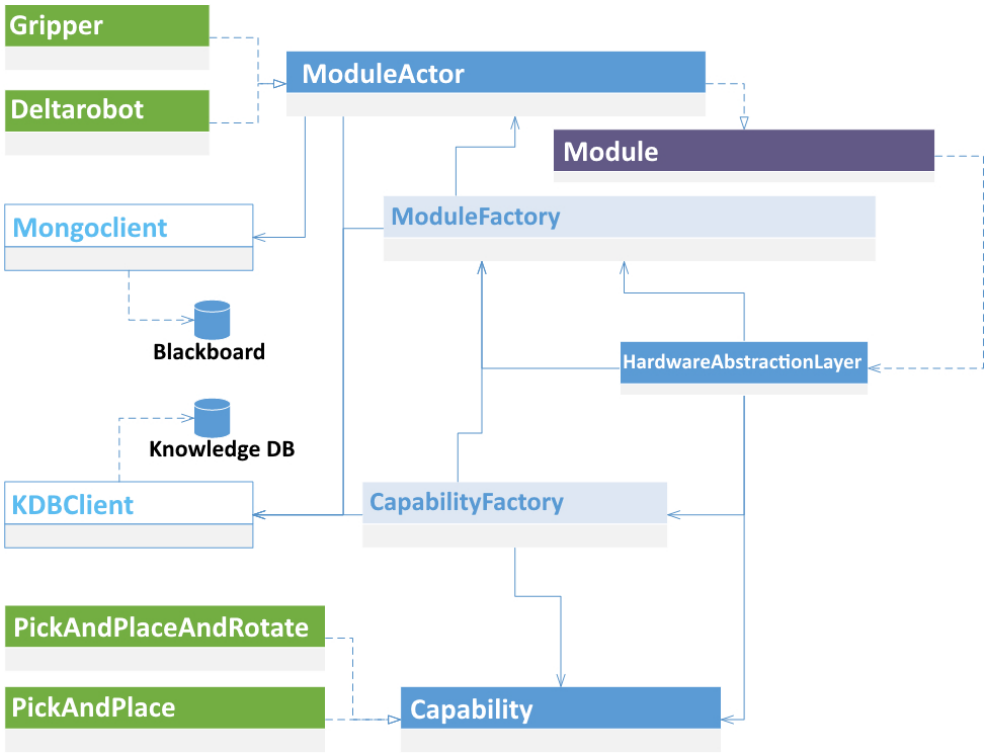


Figure 5.11: An Abstract (partial) Class Diagram of the HAL.

5.8.2 Translation and Execution Process

The entire process of how the HAL is used can be seen in Figure 5.12. The process starts with an equiplet asking for a product step that needs to be translated. The request is handled by the TranslationProcess thread, which checks what capabilities are required to perform the product step. This is performed by using the CapabilityFactory that queries the Knowledge Database (in the Figure shown as DB). These actions result in composite steps that contain several mutations, and a check of the functional module tree, mentioned before in Figure 5.10. The functional module tree is used to check if the current hardware configuration of the equiplet is capable of performing the steps. The composite step is processed step by step from the bottom of the FMT. Each module checks which mutations it can perform through the use of the capability class, which queries the knowledge database. When it is able to perform the mutation it is taken out of the composition step. A translation is successful if the composite step is 'empty' after passing through all the modules in the module tree.

If the translation is completed and all limitations are checked, the equiplet

is correctly configured and 'capable' of the task. This triggers the last step to translate the composite steps to explicit hardware steps that can be executed by the equiplet. Execution is also performed using the HAL.

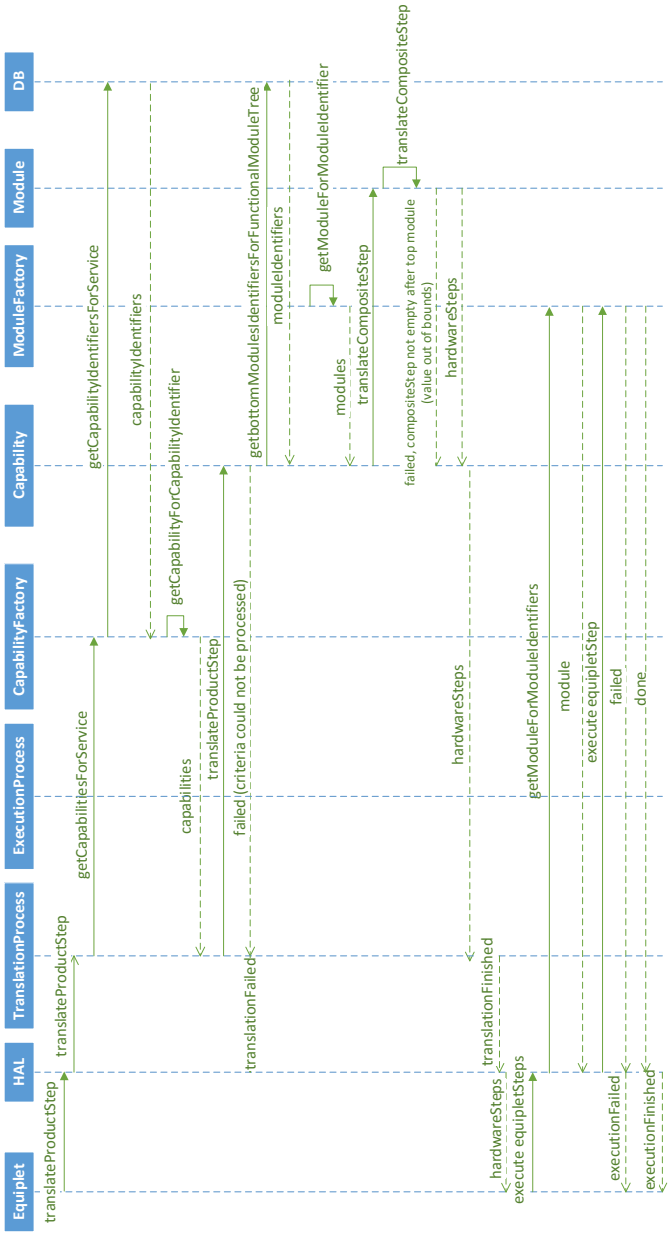


Figure 5.12: Action Diagram of the HAL.

5.9 Implementation

Reconfiguration is based on two main systems, the Hardware Abstraction Layer (HAL) and the Knowledge Database (KDB). The current section describes the implementation of these systems based on the research and resulting design.

5.9.1 Knowledge Database

To enable flexibility most systems are data driven. Hence, the Knowledge Database (KDB) is an important aspect for the reconfiguration process. The database is used for storage of semi-static information, i.e. data regarding the configuration of an equiplet, its modules, capabilities and services. At this moment each equiplet has its own KDB. However, when changing modules, data can be transferred through multiple interfaces (manually, i.e. by flash drive or over the network) between the different databases that exist within a grid.

Figure 5.13 shows the entity relation diagram of the knowledge database. The knowledge database stores information regarding the following basic entities:

Modules are the physical hardware configuration of the equiplet. The set of modules within an equiplet defines the supported capabilities and thus the supported services. Each module is either used by an equiplet or in storage. If the module is configured, it is connected to the standard mountplate at a certain coordinate. Some modules cannot be connected to the equiplet, but to another module, e.g. a gripper is connected to a robot arm. Each module must be identifiable, and each module has certain software to control and represent the module to the equiplet agent. The *Calibration* tables are used to enhance precision. Modules must be calibrated, and in order to do this, various properties must be measured and corrected. This could either be done by adjusting the hardware (for example: adjusting a potentiometer) or adjusting the software. When calibration is performed by adjusting the software, the calibration data and the calibration date must be stored. It might be possible that a combination of modules needs to be calibrated. In this case, when adjusting the software, the calibration data must be stored for that set of modules.

Composite steps consist of a composition of one or more *mutations*; mutations are an important aspect, since they describe on a modular level which standardised methods (actions) a module can perform. Mutations are very simplistically defined and must be stored in the database. Each module has a list of mutations it supports and which it requires to operate. The *required mutations* are mutations that other modules need to provide for the module

to be able to perform its own *supported mutations*. The *Equiplet* table holds a unique identifier so it can be identified within the grid. An equiplet also has a mountplate to which all the modules are attached. Modules are attached using standardised hole placements in the mountplate. These holes are laid out in a raster, with a constant distance between both horizontal and vertical holes. The dimensions of a mountplate could be defined as the number of horizontal or vertical mount holes multiplied by the distance between the horizontal or vertical holes, respectively. Due to the standardisation of the mountplate this data can be used to quickly determine the coordination of attached modules, which simplifies the reconfiguration and calibration processes. Data like schedules and position data is not stored within the KDB, but is stored directly within the equiplet agents themselves. A *Service* is a very abstract entity. It only has an identifier, which is a name. The *Capability* table is used by the equiplet to control the modules and interpret the product steps of a product. A capability consists of a specific set of mutations that are supported by the equiplet with use of the functional module tree, which was explained in subsection 5.7. A capability is considered to be of a 'type'. The type is defined as a service and is used by the product agent to identify the correct capability for a product step.

Besides these main entities the following tables are used:

The *ModuleType* table defines all hardware module types. The *Module* table defines the actual physical modules. A module could be connected to an equiplet at a certain position and has a set of properties. However, it is possible that a module is connected to another module. This is determined with the attachedToLeft and the attachedToRight fields. With the left and right fields, a tree can be constructed using the nested set model, which offers various advantages over a traditional tree. The *SupportedMutation* table shows which mutations a module supports, e.g. a delta robot can perform a move mutation and a gripper can perform a pick and a place mutation. These supported mutations are stored in the SupportedMutation table. The *CapabilityTypes* that are associated with the modules are stored in this table. Each capability type has its own piece of software. The *CapabilityTypeRequiredMutations* stores which capability type requires which other mutations. This is stored in one or multiple functional module trees. A capability cannot perform its actions if the required modules are not present. The *RosSoftware* table contains the software for the module type and the command to start the software of the module types. The *JavaSoftware* table contains the software and the classname to use for the module type in HAL. The *SupportedCalibrationMutation* describes which mutations it can support by a module within a specific state before it is calibrated. The *RequiredCalibrationMutation* table defines which states a module has to transition from and which calibration mutations are required before it can be started. Some calibration mutations are optional and

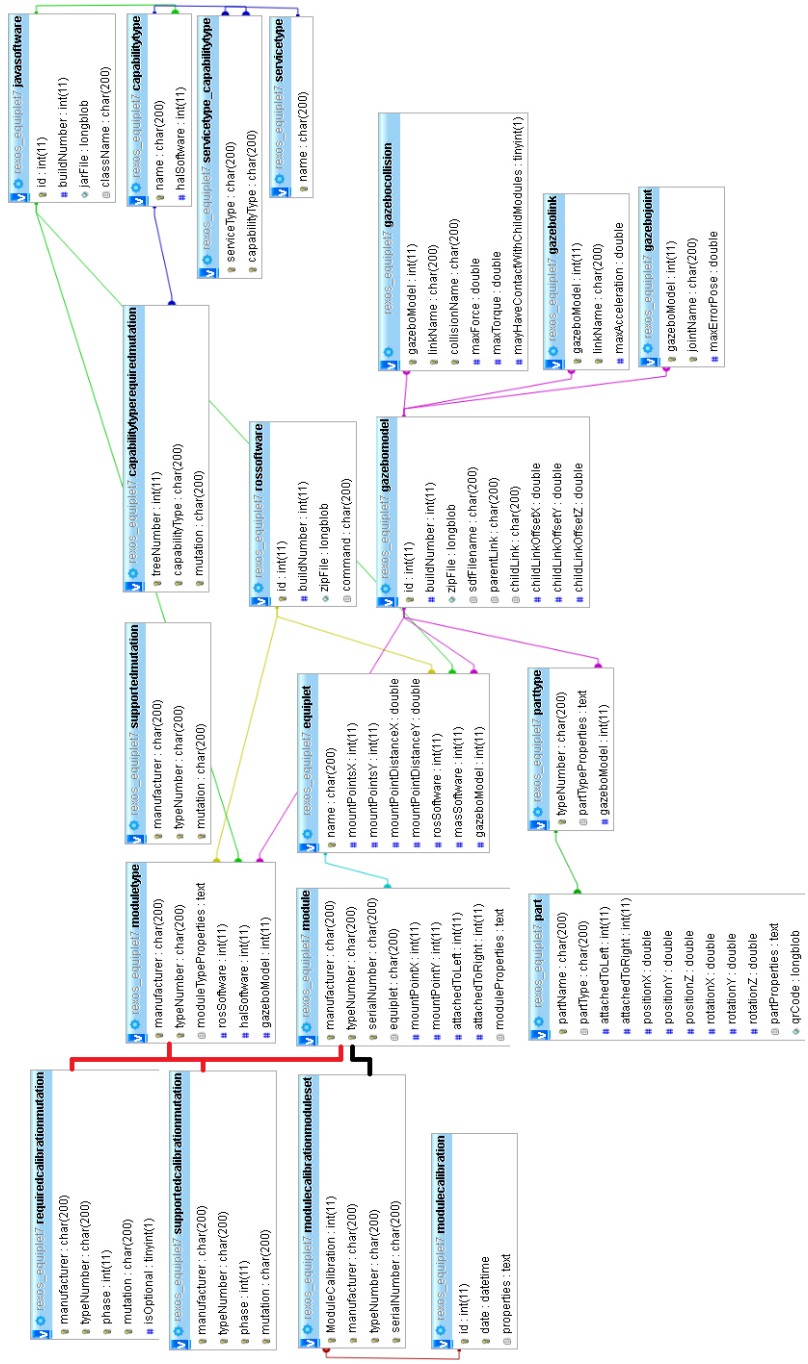


Figure 5.13: Entity Relationship Diagram of the Knowledge Database.

are thus not required for the state.

5.9.2 HAL Implementation

The HAL consists of 25+ classes and is largely written in JAVA. The HAL is a standard software object (or artefact, representing the equiplet hardware, from an environment programming orientation) and is used directly by the equiplet agent. The exact implementation is largely placed out of scope for this thesis, since it provides limited extra insight to the concepts that are important to the study. However, some internal working of the HAL will be discussed in the current subsection to give a bit of insight into the implementation.

Figure 5.14 shows the main HAL class.

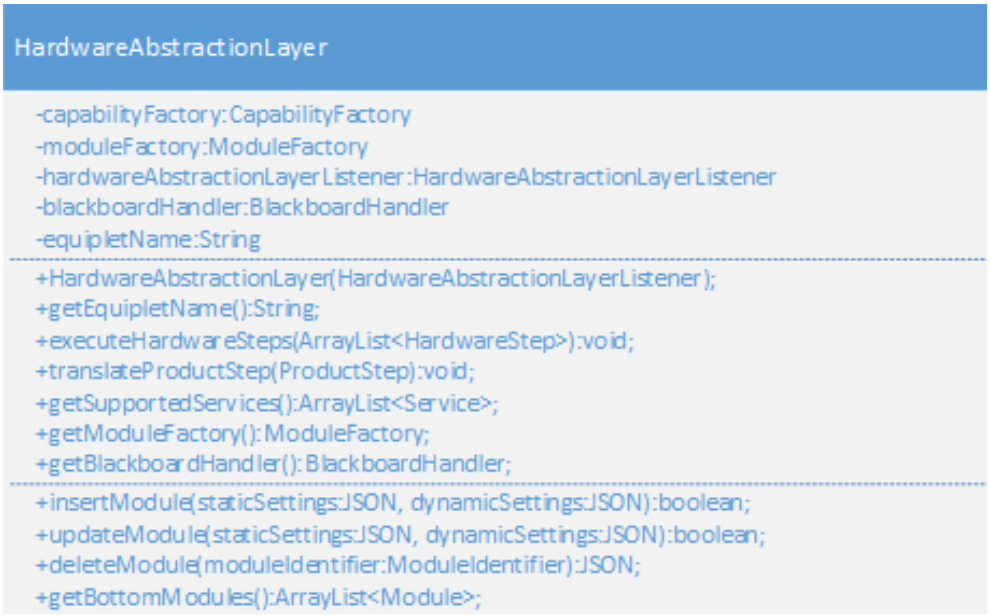


Figure 5.14: Hardware Abstraction Layer Class.

The class has three main functions:

1. Starting the translation process;
2. Execution of a hardware step;
3. (Module) Reconfiguration

Execution is performed by sending the required steps towards the hardware layer of the system (which will be discussed in the next chapter). In the current implementation they are sent through a blackboard that is used for

communication with the hardware. The translation process is performed with use of the instances of specific capability classes, which have been made by the Capability factory within the HAL, see 5.11 again for an overview of the main classes. The created capabilities are used to translate the steps. Finally, the reconfiguration is performed directly by the module factory.

5.10 Discussion

Decoupling the product manufacturing instructions, the product steps, from generic services and the hardware, creates a large degree of freedom in designing products and to utilise the reconfigurable manufacturing systems. The proposed decomposition system that translates the steps has never before been completed for industrial application. However, it has similarities with the approach of Heintz et al. (2007), which uses a knowledge processing on different levels of abstractions to bridge the sense reasoning gap for uses in autonomous UAVs.

The goal of the overall study was to design and test the capabilities of a grid, by creating fully functional prototypes of a grid that can perform a number of different services. To be able to do this, both flexibility and costs are of importance. As such, equiplets are designed to be low-cost, easily customisable, and both hardware and software systems needs to be standardised. This has also led to a standardised platform, on which a range of products can be added. Flexibility is increased further by the generic concepts of grid manufacturing, using autonomous software systems and reconfigurable hardware with a dynamic transportation system. Together with the discussed translation process opportunities are created for new business cases. This is important, since *Schild and Bussmann do mention that agent-based manufacturing systems still cope with difficult adoption by the market*, mainly because flexibility is not often a business case in and of itself (Schild and Bussmann, 2007). Therefore the approach of grid manufacturing using equiplets also focuses on specific markets, like Microelectromechanical (MEMS) manufacturing and possibly new consumer markets like 3D-printed custom products or industrial rapid prototyping.

While the current system provides a flexible basis for manufacturing, there are challenges. One is the specific software that is required for the translations between levels. To be able to design a large variety of products, using a variety of software, and also a large variety of different configurations, *it is necessary to create a large amount of classes that are able to specifically address challenges that come with this variety.* To simplify this, not only will the processes involved in the manufacturing need to be standardised, as we provided some steps for in the current chapter, but also the hardware and product

definition. Some steps are being taken in this direction. E.g. Järvenpää has recently introduced a rule-based system to automatically match products and capabilities of a manufacturing system (Järvenpää et al., 2012).

Another challenge is the distributed and autonomous architecture itself. Currently responsibilities of systems are clearly defined. However, what happens if a system, due to an error, cannot fulfil its responsibilities? Current error behaviour of equiplets involve the equiplet agent contacting its scheduled products to try to reschedule. When an equiplet suddenly becomes deactivated it might also be unable to report to the equiplet directory that it is no longer capable to fulfil its steps. Initially, this might not be a big problem, since the product will likely have several equiplets it will negotiate with. If one does not answer this will not provide any problems, since it will choose another capable equiplet. However, one can imagine these problems to grow to a scale where the equiplet directory will require a clean up of outdated data to be able to give the correct information to the product agents. These and other similar cases might require more proactive actions from other agent systems that need to be added to the architecture, increasing the complexity of the entire architecture.

While many agent-based manufacturing systems have been mentioned in the literature, they have barely been adopted by industry. The lack of practical implementation is mainly due to the large costs, low amount of examples of flexible manufacturing systems that have made it beyond the laboratory (Leitão, 2009), and a lack of standardisation (McFarlane and Bussmann, 2003). The current chapter addresses some of these problems by increasing the flexibility not only from a logistical perspective, but also by decoupling the hardware and design process from the actual services that can be provided for agile manufacturing purposes. The current chapter also mentions how the architecture is used to successfully create a heterarchical system where products can be dynamically produced. Together with the reconfiguration this provides a scalable approach to manufacturing. Perhaps **the most important part of this chapter describes the translations between production steps**, providing the ability to dynamically use a variety of RMS to be able to optimally manufacture a range of products.

The presented architecture shows promising results to aid in the future adoption of flexible agent-based control systems and provide the ability to offer new business cases to companies and create a shorter time to market for new products. Finally, *within the near future Grid Manufacturing might have the impact to create more automated product manufacturing for high-mix, low-volume products*, in contrast to manual labour.

5.11 Future Work

Current work has focused on the translations and standard infrastructure to be able to perform a variety of steps and support a small range of configurations. While these systems are now largely available, some systems still need to be improved. This includes the logistic and optimisation system. These systems will be extended to take critical decisions in the manufacturing process, e.g. it seems possible to dynamically optimise manufacturing processes in a grid, by utilising BDI agents that will try to optimally advise equiplets and products in the grid to optimally utilise the logistic systems and availability of equiplets. Possibly, it can even advise to reconfigure the grid in such a way that production lines will automatically 'emerge' in a grid when a large amount of similar product steps will need to be manufactured. In some cases it might be more efficient to have certain equiplets close to each other or, depending on the size of a batch, to create lines in a grid where the next product step will always be completed close by. The grid might be able to determine these optimisation possibilities with the use of simulations that evaluate future demand from the current active product agents and grid configuration.

5.12 Conclusion

To conclude the chapter let's restate the research sub-questions for the current chapter:

RQ2a How can the product use the machines when it has no knowledge of the manufacturing system or its (hardware) configuration? By standardising the abstract steps it is possible to use a generic method that describes the manufacturing process of any product. The equiplet also registers the services it can provide at a Directory Facilitator, which the product uses to search for an equiplet that can perform its steps. This way a product can be manufactured without knowledge of the hardware of the manufacturing machine. Hence, different machines could also possibly perform similar steps, thereby increasing flexibility and possibly efficiency.

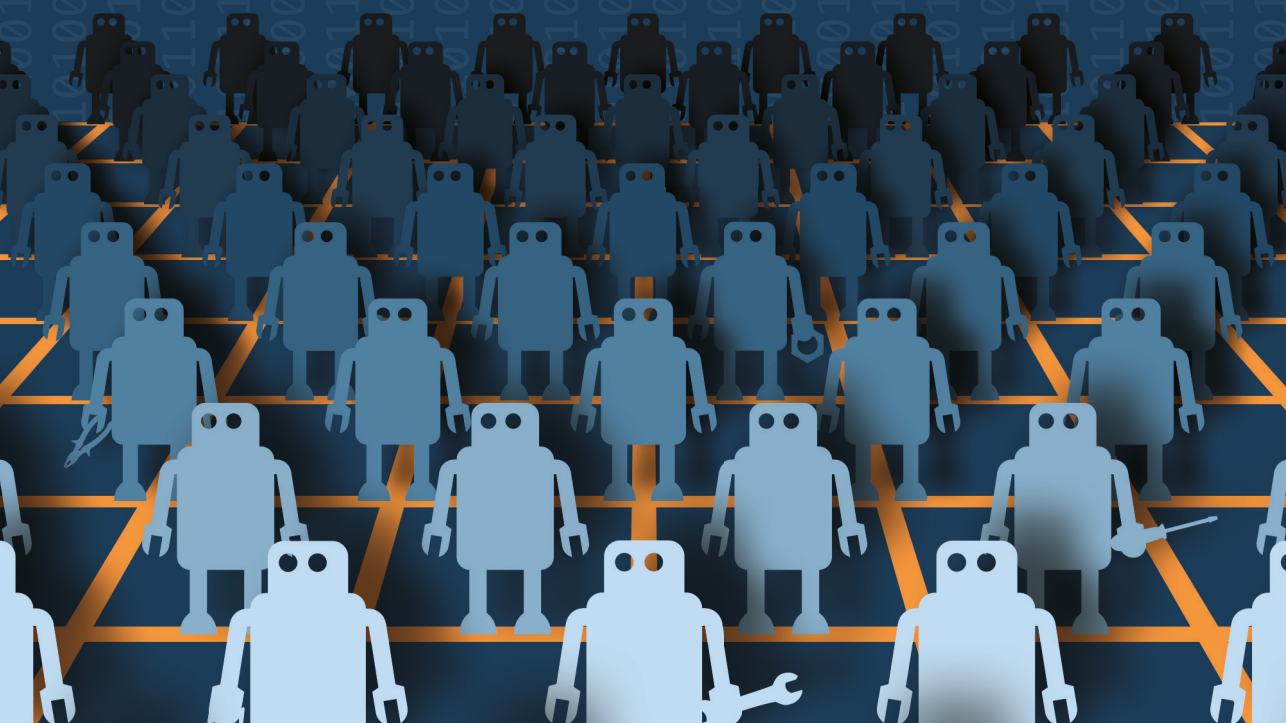
RQ2b How does the machine know which services it can provide after it has been reconfigured? The machine uses the information in the Knowledge Database to figure out its capabilities. This is done by creating the Physical module tree that provides the dependencies between modules. From here the Functional module tree is created that shows the standardised methods (mutations) that each module in the configuration can perform. Within each module the Knowledge Database shows which mutations it requires and which it supports. The database can also check per service which mutations are required. When these match, the configuration is able to perform the service.

RQ2c How can the machine control its (changing) hardware modules? The (abstract) product steps are decomposed by the HAL, which translates them step by step for the specific hardware instructions that are required for operating the equiplet where the product step is scheduled. The decomposition process makes use of the information in the HAL and Knowledge database. Hence, the translation is completely data-driven. More systems can be added by adding the new information into the Knowledge Database.

This brings us to the original main question for this chapter: *RQ2 Can reconfigurable manufacturing systems be controlled without the need to reprogram them for every new product or hardware module?* The answer to RQ2 is yes. By combining the abstract product steps and the HAL it is possible to create a system that is mainly data driven. The HAL has a number of factory classes that can create the required software modules when needed from the Knowledge Base. This opens the opportunity to add new modules, products, and capabilities to the grid without reprogramming existing and running systems.

The main contribution of the current chapter is the use of the HAL, with its data-driven translation system. Which creates a flexible system for reconfigurable manufacturing systems.

06



Architecture

“If you wish to make an
apple pie from scratch,
you must first invent the
universe.”

– Carl Sagan

Architecture

The Vision and Reconfiguration chapters lay the foundation for Grid Manufacturing, providing flexibility in finding and handling objects, and flexibility in how machines can be changed and used. Now that these foundations have been established, it is time to define an overall software platform that utilises the abilities of dynamic handling, and reconfiguration. The software architecture for Grid Manufacturing is called REXOS, which comes from **R**econfigurable **EQ**uiplet**S** **O**perating **S**ystem - REQSOS, where the QS is replaced by an X. REXOS is meant as a proof of concept architecture that will be developed and tested, based on the research nad developments discussed in this thesis.

Parts of this work have been published already in (Telgen et al., 2015b).

6.1 Problem Description

Grid Manufacturing in general should be flexible. The last chapters have shown that making use of data from different systems and dynamically using data driven systems can create this flexibility. However, flexibility also means that systems are not designed for a specific purpose and have to adapt often. This makes it harder for these systems to be optimised for performance. Yet at the same time, performance is important for control of the hardware. Hence, the problem is how to combine the flexibility on the high-level systems that cooperate with the low-level systems, i.e. hardware, which need real-time performance to run efficiently.

This chapter focuses on the platform that will provide for these problems and which will be the basis for further study in the following chapters.

6.2 Research Question

The research question will focus on the main problem of this Chapter, the creation and specification for a flexible software architecture that will provide both performance for low-level functionality, as well as flexibility for high-level functionality.

The main research question for this chapter was: RQ3 - What options are available to combine flexibility and performance for software architecture in grid manufacturing?

For the design it is important to examine a number of platforms and research how they can interface without becoming coupled. The next section

will review a number of platforms and technologies that could become the basis of the Grid Manufacturing Software Architecture. To do this we first look at one of the fundamental design properties for the architecture.

6.3 Design

As mentioned in the research question, the software architecture is expected to have two commonly conflicting parameters, that is performance and flexibility. These also follow from the functional requirements described in the Grid Manufacturing concept that were discussed in Section 3.3.3.

Basically, these can be seen as two different requirements on the highest level. When using Axiomatic Design it can be shown that a hybrid architecture could be an option of interest.

To explain axiomatic design in more detail, an important rule of Axiomatic Design was considered:

- Axiom 1: The independence Axiom - Maintain the independence of the functional requirements;
- Axiom 2: the information Axiom - Minimise the information content of the design.

These axioms indirectly show why cooperating autonomous systems like MAS lower the complexity of a design, since many functional systems can be isolated in a single entity.

As mentioned in the Problem Description, the architectural design has to be able to have high performance and intelligent behaviour. This is a common problem that has been recognised in recent literature (Heintz et al., 2007), and has t be taken into account when designing a new architecture. Based on Axiomatic Design (Suh, 1999) describes ways to 'decouple' the design parameters, i.e. how things are solved from the Functional requirements, which is to say: what they should do. Axiomatic Design calls a design uncoupled if one design parameter only affects one functional requirement. A classic example that shows this is a warm water faucet. A warm water faucet has two functions: 1. to adjust the temperature. 2. to control the flow rate of the water. In traditional systems this was solved with two design parameters: A. A cold water handle, B. A warm water handle. However, this defies the axioms of axiomatic design since each design parameter (the water handles) influences both functional parameters. Hence, a good design will decouple this, by for instance using a thermostat and flow handle which controls both aspects separately

Based on the methodology of axiomatic design this urges us to think of how to decouple the required properties of the architecture. From this perspective

it becomes clear that the need for flexible, intelligent behaviour, and therefore abstractness is in contrast with the performance. Hence, these systems could be decoupled by using different platforms for each purpose. This could be achieved by creating a separate high-level, and a low-level platform. The high-level platform will deal with unexpected behaviour and logistical matters, while the low-level platform will directly control the hardware.

Table 6.1: Simplified Design Matrix that shows the relationship between FRs and DPs.

	Low-level Platform	High-level Platform
High Performance	x	
Intelligent Behaviour		x

Table 6.1 shows the relationship between the requirements and the solution. In this case, the requirements are decoupled through the creation of a *hybrid architecture* where multiple platforms are combined to potentially yield the best of two worlds (Arkin, 1998). This in contrast to a system where one platform is used where these requirements should be combined. Note that these matrices commonly span much more details. However, in this case only the choice for a hybrid platform is examined.

This choice is fundamental for the architecture. Instead of one platform, two platforms will be used that will need to be combined for the creation of the Reconfigurable Equiplets Operating Systems (REXOS) platform.

6.4 Choice of Technology

Grid Manufacturing can be seen as a complex system where many autonomous systems have to interact. This is one of the reasons why it is important to use autonomous entities to become as flexible as possible without creating too many interdependencies that increase the overall complexity of the system. As shown in Figure 6.1, and discussed in the Concept Chapter, a multi-agent system could fit this requirement in that it offers a level of abstraction and limits the sphere of influence for an entity.

The use of a MAS seems a good option to choose as a basis for REXOS, if we investigate this further we can also look at five characteristics of a manufacturing environment (Moergestel et al., 2011):

1. Autonomy
2. Cooperative
3. Communicating

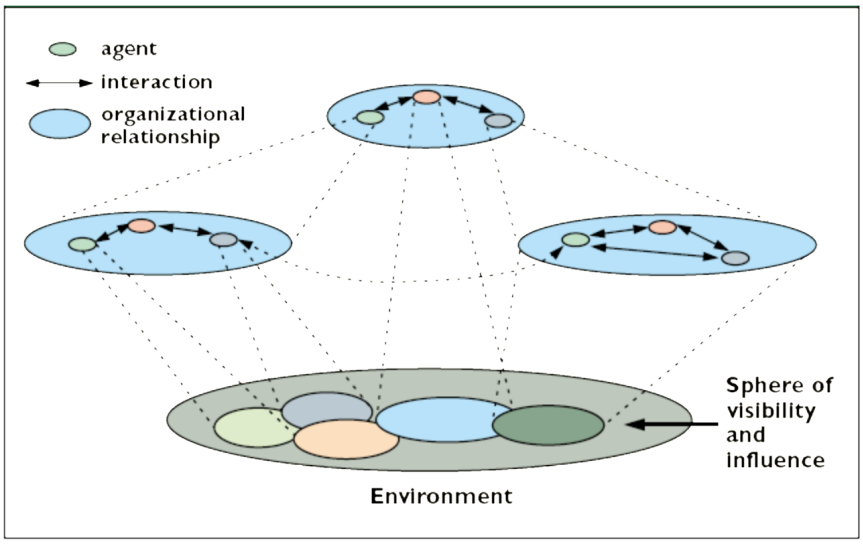


Figure 6.1: The sphere of influence within a MAS (Jennings, 2001).

4. Reactive

5. Pro-active

Together with the set of requirements for equiplets and the grid, these fit perfectly into the concept of grid manufacturing. Besides the manufacturing processes themselves, agents provide many more possibilities that are out of the scope for this study, e.g. a product agent that stays with the product to analyse its behaviour and offers problem solutions whenever possible (Moergestel et al., 2013a). The agent that can represent hardware could also be utilised to analyse efficiency and learn from the behaviour to optimise schedule times and other logistic matters.

6.5 Choice of Platforms

Even though we choose to use MAS as a basis for REXOS, this does not fulfil all requirements for grid manufacturing. MAS will provide a dynamic decision platform that will represent all systems. However, it is normally not suited for direct real-time control of hardware. Hence it is important to investigate which platform could fulfil this requirement and to research how these platforms could be successfully combined. To approach this task lets first look at a number of agent platforms.

6.5.1 Agent Platforms

The platform that is used for REXOS has to meet certain requirements as mentioned in the problem description. Five attributes also have to be satisfied, which are part of the Customer Domain:

1. The platform needs to be scalable.
2. For flexibility the platform needs to be able to change or add new agents during runtime.
3. The platform needs to be mature (for industrial application).
4. Performance needs to be sufficient to handle grid-wide logistics.
5. The platform should preferably be open-source, but also applicable for industrial use with propriety, i.e. commercial/closed software, sources.

Six agent platforms have been investigated:

- 2APL[1] (Dastani, 2008)
- JADE[2] (Bellifemine et al., 2007)
- Jadex[3] (Braubach et al., 2004)
- Madkit[4]
- Jack[5]
- Jason[6]

The choice for the agent platform has become Java Agent Development Framework (JADE), since in JADE agents can migrate, terminate and start in runtime. JADE has also been widely adopted and has an active community. While JADE has no direct support for BDI agents it can be extended to add this when necessary in the future. Currently the architecture does not force the use of BDI. JADE is also compliant with the Interoperable intelligent multi-agent systems specifications standard Foundation for Intelligent Physical Agent (FIPA). Which makes it possible to easily extend the MAS with other FIPA compliant systems.

[1] http://apapl.sourceforge.net/
[2] http://jade.tilab.com/
[3] http://sourceforge.net/projects/jadex/
[4] http://www.madkit.org/
[5] http://www.agent-software.com.au/products/jack/
[6] http://jason.sourceforge.net/wp/

6.5.2 Diverse Platforms

Besides the agent platform for the high-level systems, there is a need to combine it with other platforms to control the low-level aspects, i.e. the hardware, and satisfy the Customer Attributes and Functional requirements.

Robot Operating System (ROS) is a software framework that provides services, tools and libraries for robots (Quigley et al., 2009). The framework has extensive support for a variety of sensors and actuators and offers hardware abstraction and low-level device control. ROS is free and open-source and uses nodes as software modules that communicate with messages. Nodes can be started and stopped in runtime, making it possible to adapt software modules at any time. ROS has been created to create general purpose robot software that is robust.[7]

Robot Operating System (ROS 2.0) is currently under development and is being created to overcome some limitations of ROS 1.0, including real-time requirements and use for multiple robots.[8]

MongoDB Due to the diversity and flexibility of the grid it is difficult to define all schemas that relational databases use. MongoDB uses dynamic schemas, is cross-platform and has a document-oriented database. Hence, MongoDB can be used as a blackboard between platforms.

OpenCV Open Computer Vision can easily be integrated with ROS, it is released under the BSD license and can be used on multiple platforms. It has a focus on real-time applications and has been proven in many projects. Hence, it is logical to choose for the OpenCV library to integrate OpenCV in REXOS. The computer vision is used to identify and localise parts within the working space of the equiplet and is used for other logistic processes necessary for configuration and calibration of the systems, e.g. identification of a new gripper.

6.6 Reconfigurable Equiplets Operating System

JADE, and ROS (1.0) have been chosen as the platforms to develop the hybrid platform. The main reason for using ROS is its proven usefulness in many projects and the experience from other projects[9].

REXOS will be a distributed multi-platform system and as such will run on a number of computers. Figure 6.2 shows a simplified overview of REXOS.

The basic logistics will run on a grid server that provides a Directory Facilitator (DF), Grid Data Acquisition System and Logistic manager. The DF

[7] http://www.ros.org/about-ros/

[8] http://design.ros2.org/articles/why_ros2.html

[9] including Artemis R5-COP http://r5-cop.eu/ -last accessed 07-05-2016 and the RAAK-MKB AEROBIC project

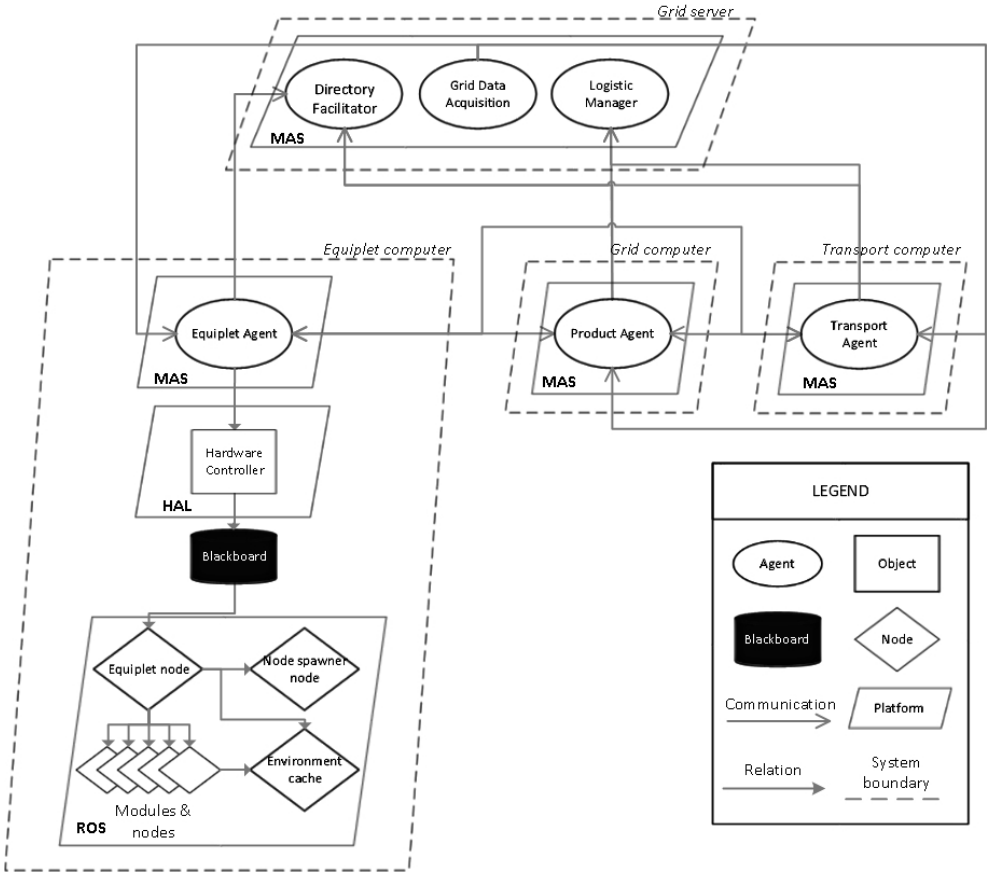


Figure 6.2: High-level software design of REXOS.

can be seen as a yellow page service that knows which equiplets are active and what services they provide for the products. The Data acquisition will be used for a possible connection to statistical and Enterprise Resource Planning (ERP) Software. The Logistic manager is mainly meant for transportation of parts and products within the grid. The Hardware Abstraction layer, extensively discussed in Chapter 5 Reconfiguration, is also shown. The HAL will be used directly by the equiplet agent and is connected through an interface to the ROS platform, which will be discussed in more detail later on.

Products will be created dynamically, usually by an application, and will then be moved to the grid (server) where they will be produced. If the product has an embedded computer the product agent will be moved to the product after it has been completed. However, it might also exist in the cloud. This way the product agent can be of value throughout the entire life-cycle of the product. This way it can provide a number of services for the owner and

others who use it, e.g. manuals, repair or recycle information (Moergestel et al., 2010).

Transport agents are responsible for the transportation of the device. Depending on the implementation this could be done in a number of ways, including Autonomous Ground Vehicles (AGV) or with (multi-directional) conveyor belts. The equiplet will be the main system in a grid and will have a number of main platforms that each consist of one or more entities:

1. The Equiplet agent
2. The Hardware Abstraction Layer (HAL)
3. The ROS layer

All these platforms will commonly reside on one computer and are embedded within the equiplet. The equiplet agent will represent the equiplet hardware and interact with the grid and the products. It will also deal with scheduling and determine its capabilities based on its configuration. When a product arrives on schedule to be manufactured it will send its product steps (Moergestel et al., 2011)to the equiplet agent, which will forward it to the Hardware Abstraction Layer (HAL). The HAL can interpret the steps and translate them to specific instructions known as 'hardware steps' that will be sent to the ROS layer to be executed.

The ROS layer consists of an equiplet node and at least one node per module that represents the hardware module. It also consists of a spawner node that is able to start new nodes when modules are reconfigured. The equiplet node will receive instructions from the HAL. For the ROS layer an *environment cache* has been created that represents the physical dynamic environment. Information that the environment cache holds is, for example, the position of products that are perceived by a computer vision or external system.

The interface between the different platforms is essential for a successful hybrid architecture. While Figure 6.2 shows a blackboard, other implementations have been developed and will be discussed in the implementation section.

6.7 Implementation

This Chapter is limited to the basic architecture of REXOS. The main focus will therefore be on the infrastructure and platforms that are required for grid manufacturing. As mentioned before in the introduction, the hardware is an important aspect to prove the feasibility of the concept. As such this section will show the implementation of the software platforms, the interfaces, but will also give an overview of the hardware that is used.

6.7.1 Middleware

Figure 6.3 shows the implementation of an example of the JADE platform for grid manufacturing, which consists of two equiplets and a grid server. JADE uses a main container that can be connected to remote containers (which are in the other equiplets). The main container holds a container table (CT) and two special agents, called the Agent Management Service (AMS) and the Directory Facilitator (DF). JADE has the ability to replicate or restore the main container to remain fully operational in case of a failure.

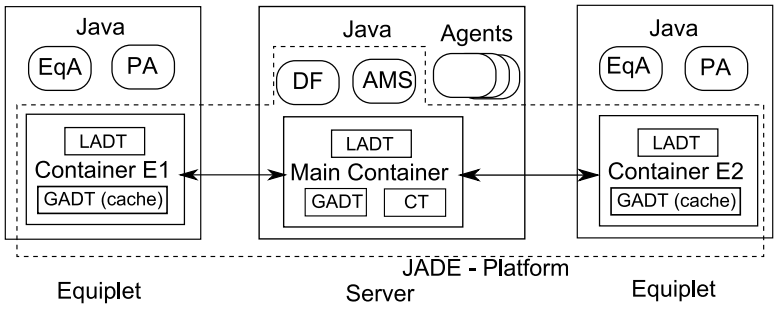


Figure 6.3: The Java Agent Development Platform.

In every container there is a Global agent descriptor table (GADT) that registers all the agents in the platform, including their status and location, and a local descriptor table (LADT). The GADT in the remote containers will be used for caching.

EqA and PA are the Equiplet and Product agent who will represent a specific product or equiplet.

6.7.2 Grid

The grid provides logistic functionality, which the autonomous equiplets can use. Based on the architecture shown in Figure 6.2 it is standard that the GRID services run on a separate server. However, since the software runs on a standard Linux system and the JADE environment can be moved or distributed in any way, the GRID functionality can be run on any computer within the network. As such it is possible to start it on a computer within an equiplet. This creates the ability to quickly setup the functionality of an equiplet without requiring a complete infrastructure.

Transport (van Moergestel et al., 2014; Moergestel et al., 2015, 2014) and other logistic systems like scheduling (van Moergestel et al., 2012) are discussed in separate research and are considered out of scope for this study. However, the references link directly to related work in this field that have been specifically applied for Grid Manufacturing.

6.7.3 Equiplets

The equiplet and its modules are specifically designed with grid manufacturing in mind. An equiplet consists of a rigid base with standard mounting points to attach modules. A standard equiplet is commonly used for assembly actions and as such typically uses 4 modules to be attached, a manipulator, gripper, vision system and a working plane. A standard equiplet stands on a rails to be easily moved and holds a standard on-board PC. Several equiplets and a number of modules have been developed and tested, e.g. Figure 6.4 shows a demo setup of 2 equiplets configured with a pick and place setup.

Figure 6.4: Equiplet demo setup.

The REXOS architecture is based on different technologies, the C++-based ROS and the JAVA-based JADE platform. Therefore, the interface between these two is an important aspect for stability, performance and, therefore, scalability issues. In reality, the first developed interface proved to have poor performance. Hence, three different interface implementations have been developed over time and tested for the best performance:

1. Blackboard - using a mongoDB database.
2. ROS bridge - using a ROS node with a websocket.
3. ROS Java - using a ROS node developed in JAVA.

The *blackboard* implementation (see Figure 6.5) uses a MongoDB database server and multiple MongoDB database clients. The HAL and ROS components each have a client, connected to the server. By enabling the replication

feature of MongoDB (usually used to keep the databases of multiple servers synchronised), the server generates an operation log, which logs all databases and collections on the server. The clients listen to this operation log using a tailable cursor, which is a specific control structure feature from MongoDB where the database does not need to requery the database each time new information is added. The tailable cursor will track the opened document and return new data that is added. This enables the clients to communicate with each other via the server without having to periodically query the server.

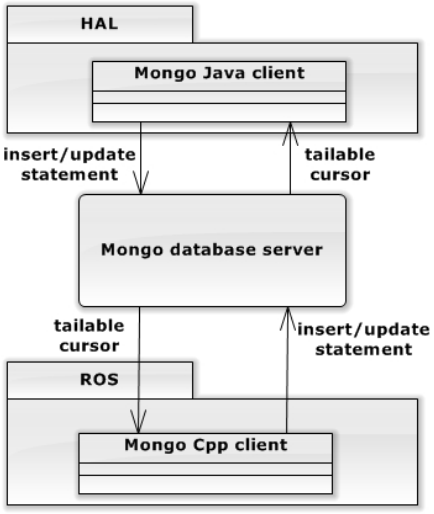


Figure 6.5: ROS HAL interface using a blackboard.

The *ROS bridge* implementation (see Figure 6.6) uses a ROS node, which acts as proxy between the HAL and ROS components. The bridge is written in Python and is designed for flexible integration of ROS in other non-C++ systems. The bridge acts as a websocket server to the outside (for REXOS this is the HAL component) and acts as a standard ROS node to the inside (for REXOS this is the ROS component). Because the bridge acts as a standard ROS node, the ROS component of REXOS can use standard ROS communication methods and messages, reducing the complexity of the interface.

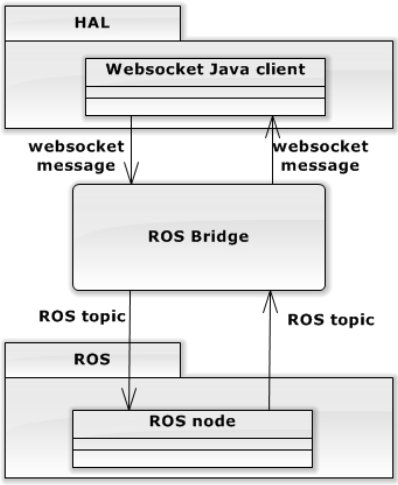


Figure 6.6: ROS-HAL interface using a ROS bridge.

The *ROS Java* implementation (see Figure 6.7) uses the rosjava_core library[10] to communicate between the HAL and ROS components. ROS is currently available for C++ and Python. Rosjava_core is an attempt to make ROS available for Java. It implements the internal ROS infrastructure including time synchronisation, namespace resolving, topic and service advertising, and communication methods.

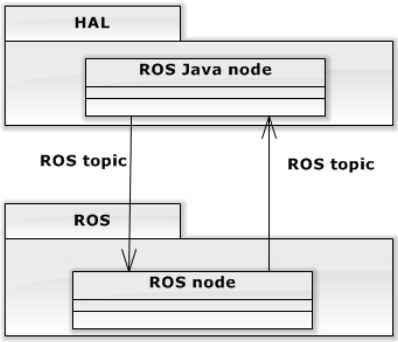


Figure 6.7: ROS HAL interface using a ROS JAVA node.

6.7.4 Modules

Modules are not designed specifically for a product, which enables the equiplets to offer generic services to a variety of products. For this purpose a compo-

[10] https://github.com/rosjava/rosjava_core

nent off the shelf (COTS) strategy is adopted together with modules that are specifically designed for grid manufacturing using equiplets, and which are developed using product family engineering. As shown in Figure 6.8, Product Family Engineering uses common parts for as many different modules as possible.

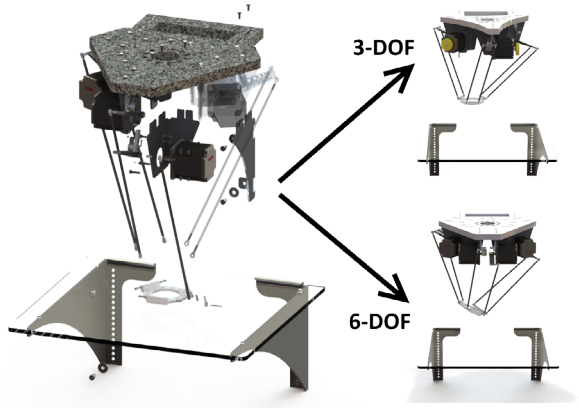


Figure 6.8: Two different parallel manipulators with 3 and 6 Degrees of Freedom using as many identical components as possible.

Currently, a number of modules have been developed specifically for grid manufacturing, we discuss six:

Delta Robot is a parallel manipulator where three actuators are located on the base, and where arms made of light composite material are used to move parts. All moving parts have a small inertia, which allows for very high speed and accelerations.

Figure 6.9 shows the schematics of the deltarobot that is specifically designed to be used for equiplets. The end effector is designed in such a way that grippers can easily be changed using a precise clicking system with magnets. Many components of the delta robot are also manufactured using additive manufacturing, making it easy to customise or produce parts for the modules on demand.

The Delta Robot uses three actuators[11] that are controlled using motor controllers[12]. The controllers are directly accessed from the respective ROS module node. Figure 6.10 shows the steppermotor class, which uses a modbus interface. The modbus interface is offered by a generic InputOutput class that is implemented by the InputOutputModBusRtuController class. This class has been created for easy reuse throughout the system and is implemented by all RTU modbus implementations.

[11] Oriental Motors PK566PMB
[12] Oriental Motors CRD514-KD

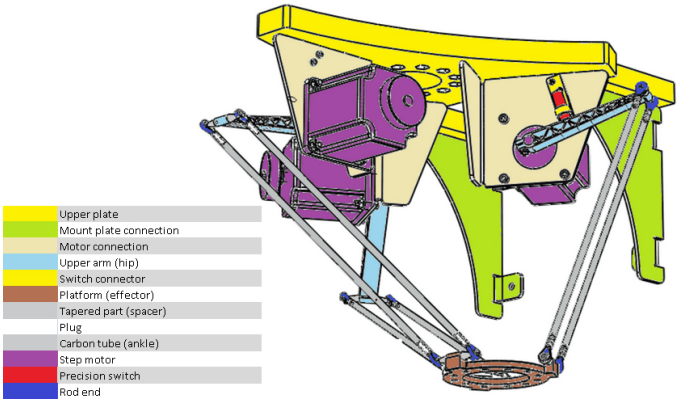


Figure 6.9: Delta robot Hardware Component Schematics.

Since equiplets are not specifically designed for a product it is important to have as much flexibility as possible to service a larger variety of product types. For this purpose, the Delta Robot design with 3 degrees of freedom was adapted to an inverted ***Stewart Gough*** platform, which uses 6 motors to get a limited 6 degrees of freedom. Figure 6.11 shows this 6 DOF parallel manipulator module. This module is very useful since products can arrive at the equiplets in any orientation.

This specific ***Gripper*** module is controlled using modbus over TCP. This is performed by an inline bus coupler[13] that is accessed from the respective gripper node. Most types of grippers that are currently used work with a pneumatic system to move an effector or create a vacuum to pick up small parts.

The ***Vision Module*** is of high importance within grid manufacturing. The product location will usually not be preprogrammed and as such must be detected dynamically in real-time. This is done by the Vision module that uses an OpenCV-based detection system to determine either the location of a product, or the location of a tray or other transport device that has the knowledge of the location of the product relative to itself. The location data found by the Vision Module will be delivered to the environment cache so that other modules can easily access it. The standard vision module holds a number of algorithms to deal with a variety of situations. It can automatically calibrate its lenses and has a built-in correction and balance system to deal with differences in lighting and distortions.

The ***Work plane*** is the area where a product, crate or vehicle that carries a product is located. Many equiplets use transparent working planes such that computer vision systems can be used to localise the specific parts. If a product

[13]Phoenix contact IL ETH BK DI8 DO4

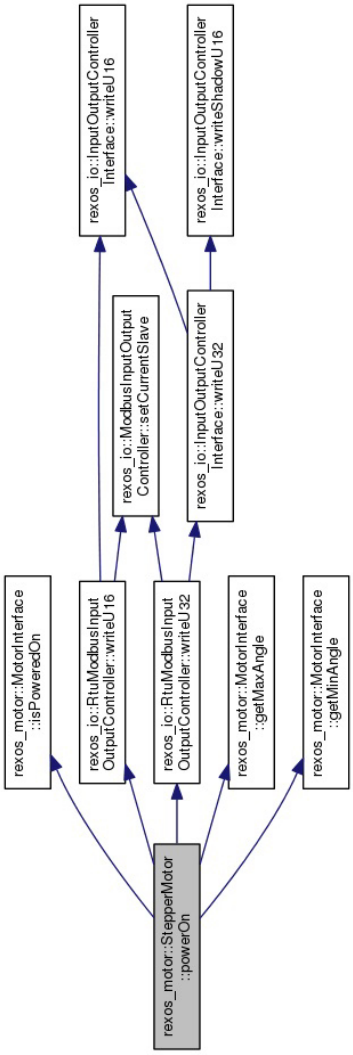


Figure 6.10: The call/inheritance graph for the motor controller.

is placed on a crate or cart the product agent can usually infer the location based on its own position as seen by the vision system. While the working plane has no actuators or sensors itself, it is still seen as a module and has a ROS node that represents it. The ROS node is used to calculate its own position based on where it has been attached, its own known specifications and calibrations using vision and markers that can be placed on the working plane.

The ***Additive Manufacturing module*** can be used for a wide range of tasks. Figure 6.14 shows the 3D printer module that can print any object, e.g.

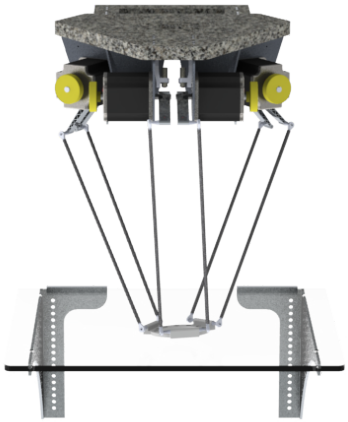


Figure 6.11: An adaptation to the deltarobot, which uses 6 motors as an inverted Stewart Gough Platform to create more flexibility.

Figure 6.12: Gripper call / inheritance graph.

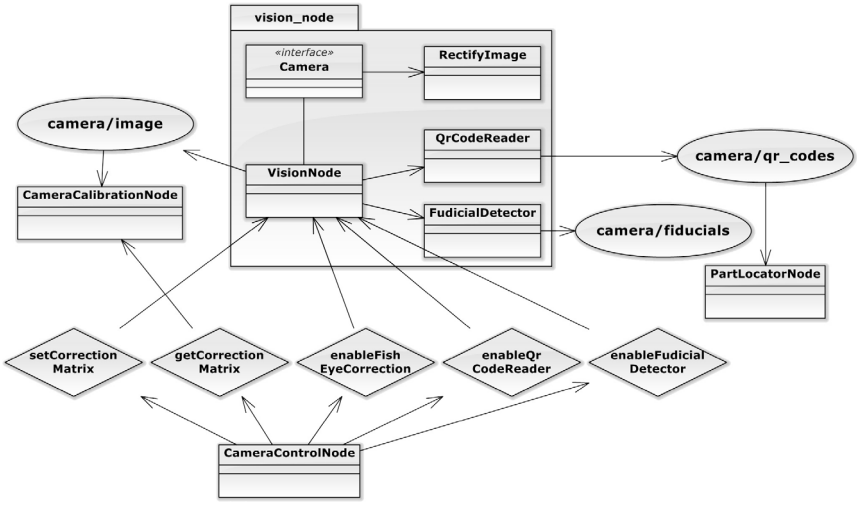


Figure 6.13: Vision system.

casings or buttons for a unique customised internet radio. This module is an

important asset for grid manufacturing since it makes it possible to create a variety of items not only for custom products, but also for the modules itself.

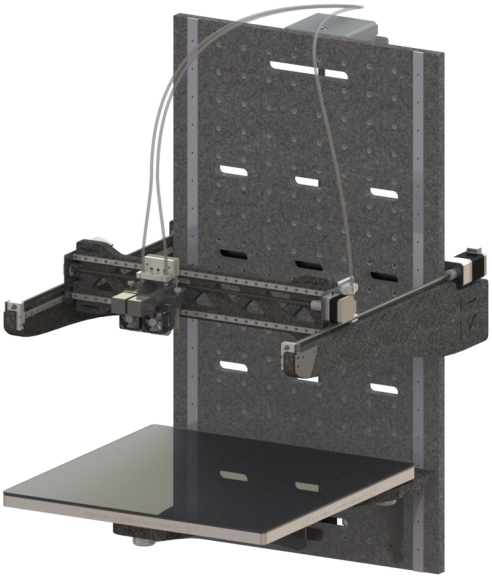


Figure 6.14: 3D printer module.

6.7.5 Basic Operation

While not all services of a grid are important for this Chapter it is relevant to show the basic operation of a grid. Figure 6.15 shows the normal operation when manufacturing. The sequence is implemented in the following way:

1. An equiplet agent is aware of the capabilities based on the modules it has configured.
2. The equiplet agent registers its service at the Directory Facilitator (DF) that acts as a 'Yellow Page' service for the Product agents.
3. When a product agent is initialised it has a number of product steps that describe how it needs to be manufactured, the product agent queries the DF to find the services that could potentially perform the steps.
4. The product agent uses the list of equiplets it has received from the DF to inquire the equiplets if they can match the specific schedule.
5. If the schedule can be met, the product agent also inquires if it can meet its specific detailed criteria that the product may require for the product step to be performed adequately.

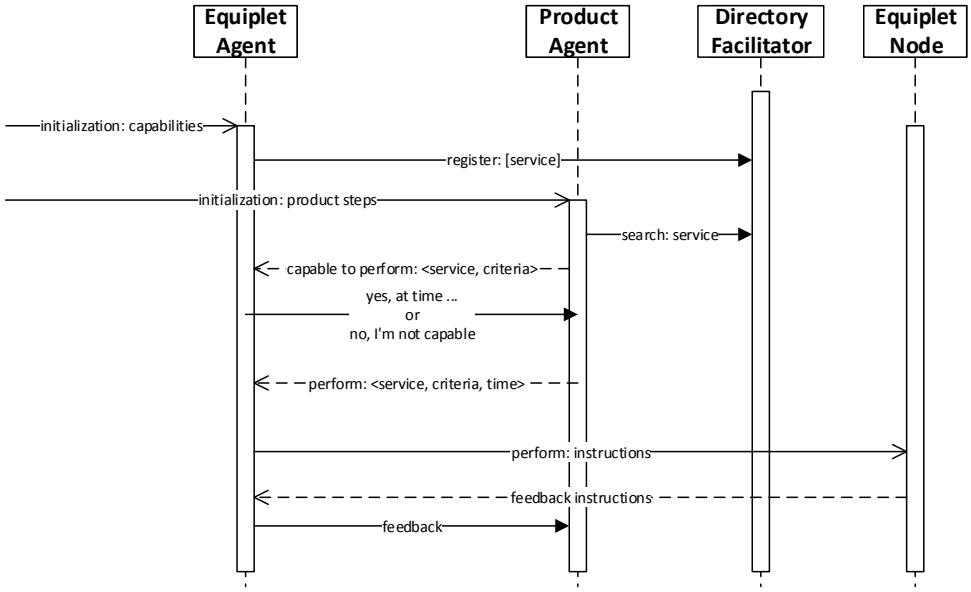


Figure 6.15: REXOS service.

6. When all criteria are met and the product arrives on schedule at the equiplet it will send its instructions on how to perform the steps to the equiplet.

7. The equiplet will translate the steps to its specific hardware and send it to the equiplet node in ROS to control the hardware and perform the specific step.

8. When done the equiplet agent will inform the status to the product agent.

6.8 Evaluation and Performance

The next step will be to evaluate the performance and scalability by performing a number of tests. These have been split in two types:

1. Synthetic test - to test the individual systems and communication latencies during load (Telgen et al., 2013b).

2. Full testing in simulation mode - to test realistic cases using the entire architecture.

6.8.1 Synthetic testing

First, a standard equiplet setup has been created that uses a ROS/JADE infrastructure connected by a MongoDB blackboard, see Figure 6.16.

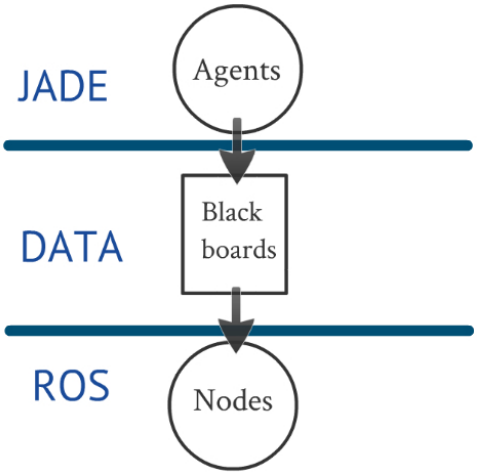


Figure 6.16: Synthetic test setup.

Three tests were performed:

1. Node-to-Node communication over ROS.
2. Agent-to-Agent communication using JADE.
3. A pick and place case utilising all layers.

For all three test cases 10,000 messages will be sent, where full round time including a response message is measured. Results are shown in Figure 6.17, which uses a trend line over 5 periods. The message is an instruction that contains a JavaScript Object Notation (JSON) object that holds a target, ID, instruction data and parameters.

The results show that ROS-to-ROS and JADE-to-JADE performance is much better than when both are combined using a blackboard. Hence, different interfaces were required to be investigated to handle the communication between the ROS and agent layer. This was performed using the simulated test system.

6.8.2 Simulated Testing

This section evaluates the performance of the entire architecture using a full simulation of the system. The sources are identical to a real runtime situation,

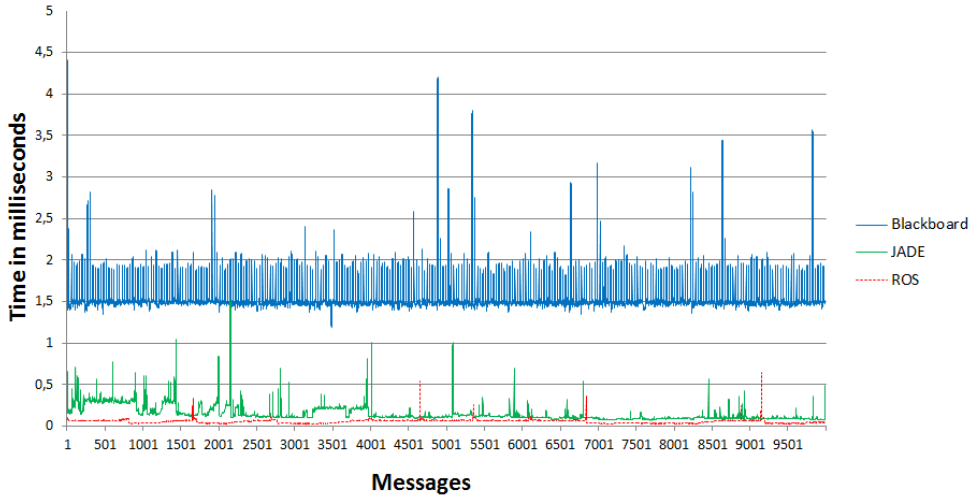


Figure 6.17: Synthetic test of ROS-to-ROS communication (bottom), Agent-to-Agent communication (middle), and the pick and place test (top).

only the hardware responses are being simulated. The most important aspect of this test is the different interface implementations that connect the (C++-based) ROS and (JAVA-based) JADE platform.

To determine the best implementation for the interface between HAL and ROS, every implementation was tested. The test has been performed using custom-written software and measures the time required to communicate from node A to node B and back to A. Node B will respond immediately. Node A is always a ROS node, while node B is either a regular ROS node, a ROS Java node, or a Java ROS bridge listener. The only exception are the blackboard measurements. Because the blackboard implementation does not use the ROS infrastructure, measuring the latency using ROS is not an accurate measurement. Instead the time required to communicate from A to the MongoDB server and back to A is measured. Because this gives an unfair discrepancy in the measurement (the relevant case is to transmit a message from A to B and receive a response from B), the blackboard latencies have been multiplied by two to compensate for the message having to be sent twice (first from A to MongoDB and then from A to B).

The idle equiplets are equiplets that have been started but are not performing any tasks. The busy EQs are equiplets executing a hardware step every 1 second and measuring data every 10ms. The measurements have been determined using 100,000 samples.

The average latency as seen in Figure 6.18 has been measured with 10 and 50 active equiplets. At the time 50 equiplets were seen as a standard

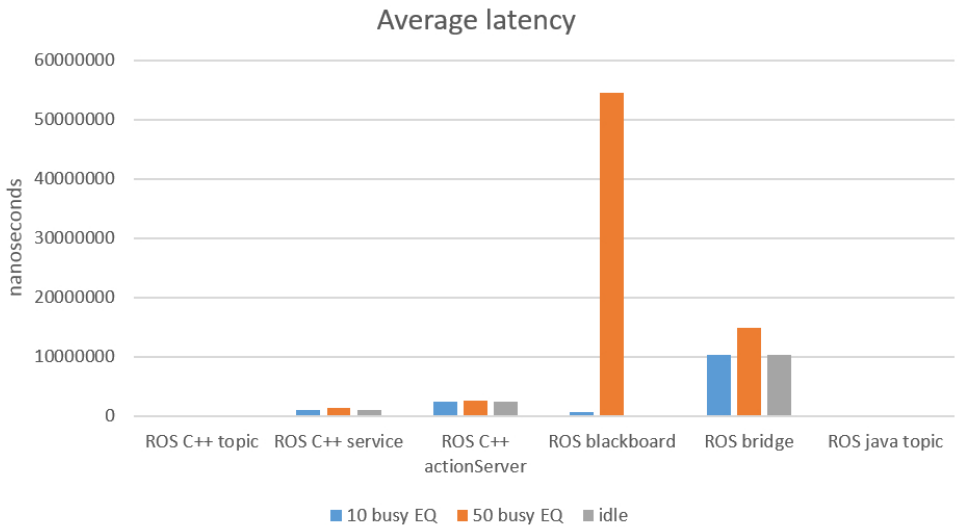


Figure 6.18: Average latency of the different interfaces that have been developed.

grid size since it would encompass enough generic services to manufacture a complete product. If more capacity was required multiple grids could than be used in parallel. Other scenarios have also been measured, but produced less relevant data. The ROS C++ topic, service, and action servers used are native ROS communication methods and act as a reference. They are not actual implementations of the interface. The data shows that the average latency for the ROS C++ topics is the lowest. The ROS C++ service and ROS C++ actionServer also have a low average latency. The blackboard implementation has a low base latency, but scales very poorly.

During the test it became clear that the blackboard implementation has a specific point after which the latency increases spectacularly. This might be caused by a connection pool in the MongoDB server running out, resulting in other connections having to wait. The ROS bridge has a very high base latency but scales much better. In all the other scenarios the base latency is also approximately 10,000,000 nanoseconds (equals 10 milliseconds). This suggests that the ROS bridge uses a periodical poll mechanism. The ROS Java topic has very low base latency and seems to scale excellently.

Figure 6.19 shows the consistency of the latency of an implementation of the interface. This shows how reliable the interface is when it comes to consistent behaviour. The average deviation matches the average latency in that once again the ROS C++ topic, ROS C++ service and ROS C++ action server perform very well, while the ROS blackboard scales poorly. The ROS

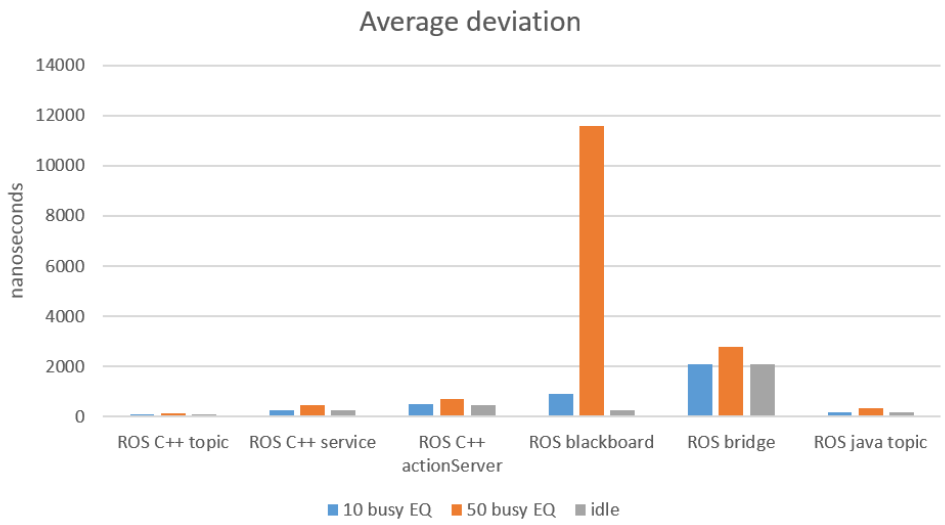


Figure 6.19: Average deviation of the latencies - note the different scale compared with the average latency.

bridge has a quite high, but steady deviation.

6.9 Conclusion

The Chapter takes an all-embracing approach to self-organising, reconfigurable autonomous manufacturing systems. The goal behind this is to provide a basis for a practical implementation by combining new technologies as a staging ground for new manufacturing methodologies based on the industry 4.0 principles that will boost the adoption by industry. Hence, this includes the development of hardware, the use of system and software engineering principles and integration of the newest hardware designing techniques. It also builds on current advances in software by using distributed systems and combining them in a hybrid architecture. The hope is that this leads to solutions that prove to industry that the newest technology is becoming more and more suitable for true mass adoption. This is done by investigating the current state of technology, analysing the requirements and new developments in smart industry and trying to encompass this in the concept of 'grid manufacturing' that consists of both a hardware platform, i.e. the equiplets, and a software platform, i.e. REXOS.

The main research question, i.e. *RQ3 - What options are available to combine flexibility and performance for software architecture in grid manufacturing*, was intended to show that grid manufacturing, based on the REXOS

platform, can combine low-level performance and flexibility using intelligent behaviour. The choice to combine two platforms is supported by the axiomatic design methodology, which strongly asks to decouple the requirements from the design parameters. The results give insights into how both JADE and ROS can best be interfaced using the JAVA ROS node that acts as a wrapper for the messages from JADE towards the ROS platform. It also shows that the interface between these platforms is crucial to get a scalable platform, by demonstrating that the (originally developed) Blackboard interface was severely lowering the performance when 50 or more equiplets were used. Additionally, the chapter shows the functionality that REXOS and the equiplets can provide.

This Chapter also evaluates the concept of grid manufacturing in general, taking design techniques and hardware into account. The equiplet platform in general and the modules specifically were designed using a low-cost strategy where equiplets can easily be reconfigured, providing a high utilisation to a minimal cost. This was done by combining product family engineering with the use of many standard components. When specific components have to be made they are commonly designed in such a way that equiplets can produce them themselves, for example by using 3D-printed parts.

More generally, this Chapter makes it clear that it is essential to take an applied approach to solve these problems. The industry will require working proof of concepts that not only tackle the theory but also the practical problems that occur when working with complex systems such as the ones demonstrated in this Chapter. The development and testing of all these systems have required a large amount of work but also add to the validity, and therefore, usefulness for industry.

This research provides a number of insights:

1. ROS and MAS can be effectively combined - which decouples the performance and intelligence gap.

    (a) The MAS provides the abstractness to deal with the dynamics that are required for self-organising systems.

    (b) ROS gives the performance and tools to effectively develop a large range of control systems that can be reconfigured.

    (c) The choice to specifically combine JADE and ROS seems to be effective.

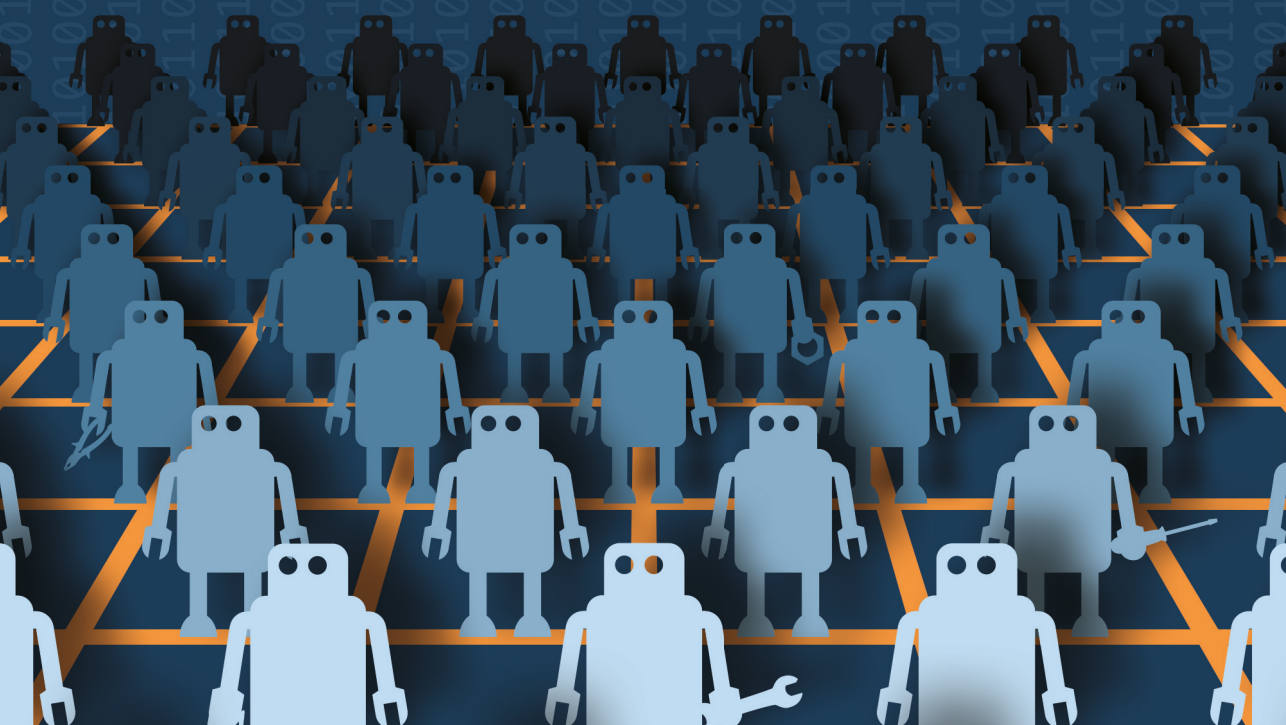
2. The autonomous nature of both platforms makes it possible to adapt part of the systems during runtime, which is an important aspect when considering reconfigurability.

3. The use of autonomous systems makes it easier to lower interdependence between functionalities, creating a decoupled design, which lowers overall complexity.

4. Combining different platforms like MAS and ROS have a high potential for industry.

The combination of the requirements and propositions gives fuel to new research in more practical problems that are fundamental for smart industry; in future work both the aspects of (automatic) reconfiguration and dynamic (safety) system behaviour will also be discussed in more detail. However, the proposition as mentioned in the research approach section seems feasible.

07



System Behaviour

“A ship in port is safe, but that’s not what ships are built for.”

—Grace Hopper

System Behaviour

Classic manufacturing systems work with completely predefined states that define exactly what behaviour the machine can have. Each state contains the steps that defines which actions it will take. This guarantees that a machine cannot start to move or behave in an unexpected way that could be a risk to itself or its surroundings. This behaviour is also clearly defined even for Flexible Manufacturing Systems, e.g. a milling (CNC) machine. When a program is loaded for a CNC sequence, the machine calculates the time it will take to perform the entire program. When pressing the start button, the access to the moving parts are blocked and the machine will go to the run state until the entire sequence is completed. When finished it will go to a safe state so that nothing will move and an operator or mechanic can take out the CNC parts without risk.

The Chapter System Behaviour in this context encompasses exactly these two aspects. First the states and behaviours of the machine, and second the safety aspect.

Parts of this chapter has been published before in the following work: (Telgen et al., 2013a).

7.1 Problem Description

Manufacturing systems should be reconfigurable and autonomous, and work in a dynamic 'chaotic' environment. This creates a challenge for the behaviour of these systems. It makes it more difficult to predict when systems are running or not, e.g. in a classic system the machine is commonly 'safe' (it cannot move) or it is 'running' (the start button has been pressed and it starts to act). With systems like equiplets these states cannot be defined as clearly. Possibly the system is ready to start a pick and place action. However, the product did not arrive yet and it is waiting for the camera to detect it coming into range. Since the product and equiplet agents are in control the state could suddenly change.

Besides the state itself, a second problem is present: There is a possibility that the startup/shutdown/error behaviour of these systems is dependent on the configuration and as such it is essential that a system is in place that defines the behaviour and guarantees safe use of the manufacturing systems.

A third problem is also introduced through the reconfigurable aspect. In classic machines the dimensions and products were clearly predefined and as

such could be preprogrammed so the machine knows the dimensions and where it could move safely. The configuration aspect changes this, since the machine can change its modules and works with a whole range of products that are unknown. Hence, it is much more difficult to predict how it moves and if these moves are safe, i.e. it should not be able to hit itself or any products by accident.

This brings us to the research questions for this Chapter:

7.2 Additional Research Questions

The main question is: RQ4 - What risks are introduced due to the reconfigurable and dynamic behaviour and how can they be mitigated? The main question leads to three sub questions:

RQ4a How can the status be clearly identified and shown when using a diversity of reconfigurable modules?

RQ4b Can a classic state machine that is used for classic machines be adapted for reconfigurable machines that act in a dynamic environment?

RQ4c How can we guarantee physical safeties when reconfiguring modules with different physical aspects?

Grid manufacturing is challenging because it has two large unknown variables: both the machine configuration and the products are unknown. To mitigate this, it is important to create a standardised state machine that describes the behaviour of the machine, and to be able to analyse if the machine can be safely used. To do this, these two aspects will be discussed separately in the two main sections of this Chapter:

Section 7.3 The Software Control System.

Section 7.4 Physical Safety.

7.3 Software Control

To minimise the potential for risk it is important to create a standardised set of states that a machine can be in. These states describe what actions a machine can perform, i.e. how it is controlled, and at what time and state an action can be taken. This will help in understanding the behaviour of the machine and as such increase its safety.

7.3.1 MAST

MAST stands for MAchine STates. It is a variation on a diversity of State machines that are used in machines and is specifically designed for use with reconfigurable machines. MAST can be seen as a part of a Supervisory Control And Data Acquisition (SCADA) system or Decentralised Control System (DCS) within an Equiplet. While most systems in the grid are not hierarchical, the equiplet's hardware and the software that does the low-level control are hierarchical, since they have physical dependencies and are controlled directly by an equiplet agent. To elaborate this point the differences in control systems is discussed.

7.3.2 Related Work

This chapter shows two perspectives of safety and control for manufacturing systems. How these (distributed) systems are organised is an important aspect of the system. In the literature there are many references to distributed control systems, even specifically for distributed production systems and agent technology (Trentesaux, 2009; Khalgui and Mosbahi, 2010; Cho and Prabhu, 2007; Barata et al., 2008; Wang et al., 2009). However, while agent technology and distributed control are often explicitly mentioned, distributed is often used in the context of (dynamic) distribution of resourcesTrentesaux (2009). This is clearly the case in Barata, who also refers to a true configurable system using a multi-agent-based control system. He also explicitly shows the states in such a system. However, he still uses a centralised and hierarchical control system (Barata et al., 2008).

Trentesaux discusses this topic extensively and even classifies control system in three classes (Trentesaux, 2009). Class 1: Centralised, Class 2: Semi-hierarchical, Class 3: Heterarchical. While class 1 and 3 are self-explanatory, class 2 is defined as a heterarchical system with a hierarchical layer. Wang also clearly focuses on a distributed approach: "a distributed and adaptive approach is considered suitable for handling the dynamic situation." However, looking at his solution he uses (distributed) real-time information to create a (centralised) cyber workspace where resources are updated by all distributed systems. While this system is truly distributed from a status and sensor perspective, it is still centralised and hierarchical (Wang et al., 2009).

7.3.3 MAST Problems

The configuration of classical SCADA systems are stable, and systems have a single purpose. The control system is usually centralised, i.e. hierarchical, or at times distributed in such a way that predefined systems have a specific task. Data is distributed between these systems, which is commonly done

over a local area network (LAN). However, a grid is reconfigurable by itself; equiplets can be added while other systems are still active. The grid uses a heterarchical approach, where the product directly negotiates with an equiplet to perform the next production step. To make this flexibility possible, the grid does not use a centralised control system. One of the biggest differences this introduces, is that equiplets are activated on demand by autonomous systems, i.e. the product agents. Besides the safety and dynamic aspects, the distributed nature with different autonomous systems also requires another approach than common SCADA systems.

7.3.4 Predictable Behaviour

Predictable behaviour is very important for safety, which is the most important aspect when working with machines and control systems. In classic systems, states were strictly defined. Because of the single-purpose and connected systems in a manufacturing line, the behaviour of the machine was known *a priori*. However, the dynamic and distributed nature of grid manufacturing implies that systems can start and stop unexpectedly, based on actions of the product agent or when a system needs to be reconfigured. Error behaviour can also be different since the autonomous equiplets are less dependent on each other. The reconfigurable aspects of equiplets also introduces new problems. The overall state of one system can depend on an undefined number of configurations that might make the system less predictable. Hence, it is harder to predict safety.

To summarize, this leads us to three subquestions:

RQ4d How can you better define the states that represent the autonomous systems?

RQ4e How do you handle safety aspects in these systems?

RQ4f How can error behaviour be standardised when using reconfigurable systems?

7.3.5 Proposal

To solve these problems we introduce several concepts that define a representation of all systems concerning safety, with a state on several levels. Standardisation of this representation of the hardware is an important aspect to be able to create compatibility between the reconfigurable systems. This requires a clearly defined architecture.

In common practice the SCADA system is hierarchical in nature. However, while our system does provide a clear hierarchical architecture, the hierarchy only indirectly influences the hardware systems. The production process

remains heterarchical, i.e. which equiplet will perform what action is determined only by cooperative negotiation. This makes it possible to provide the maximum flexibility, which is realised by using autonomous systems. This in contrast to other reconfigurable systems, like (Endsley et al., 2006) who looks at module finite state machines in a reconfigurable manufacturing system. In their approach systems are modular and reconfigurable, but the control is still hierarchical in nature and determined by controllers on different levels.

To explain the proposed architecture in grid manufacturing, we look at the presentation of a classical and grid manufacturing system.

7.3.6 Hardware Representation

From a hardware perspective, a classic manufacturing line hierarchy is set up in four layers: Line, Cell, Module, Device (Puik et al., 2013a), see Figure 7.1.

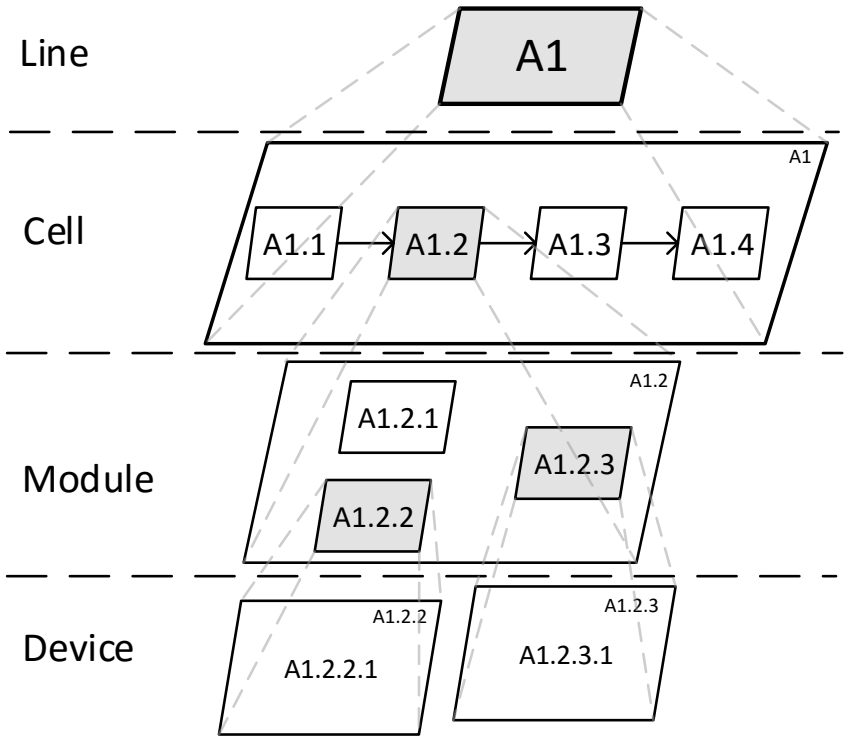


Figure 7.1: Classic line manufacturing hierarchy - LCMD - line, cell, module, device.

In this system, a line consists of multiple production cells, where each cell performs a specific operation. Products travel through the line where they are

manufactured step by step by cells, each cell conducts one specific product step. The steps are performed by (hardware) modules, which are directly controlled by the cell. Modules are specific systems that can individually receive instructions, e.g. a motor controller for a pick and place machine or Gripper. The modules are made up by devices, which are strictly hardware, like sensors and actuators that are controlled (or used in case of a sensor) by electronics.

Figure 7.2 shows the Grid Equiplet Module (GEM) architecture that is inherent to Grid Manufacturing (GM). The Grid only provides generic logistical services. In GEM every equiplet is on the same level. In GM all high-level systems have autonomous software counterparts based on the Cyber-Physical philosophy. However, this is not the case for devices and as such they are not part of the GEM architecture. Since devices have no (autonomous) software component, they are considered 'stateless'. Even if they have a state, e.g. in case of a toggle switch, the state cannot be checked. Hence, the lowest hardware state that is represented in software is that of the module. An error that can be detected at the (lower) device level, will influence the module state.

Another difference in GM is that since the machines can be multi-purpose, the hardware state representation at the module level will only influence the equiplet state when the module is actually necessary for a service that is active at that moment. Finally, the equiplets can be completely autonomous towards the grid.

All control of the hardware is performed by the modules, all higher systems on the grid and equiplet level exist only in software. Because of this the state of the hardware, the module state, is the most important state.

7.3.7 Module States

The module state (MOST) is the direct representation of the hardware of a module. Each module has its unique state. The states are divided between the normal states of the module and transitional states. A transitional state is in definition temporary, and will only last until all actions have been taken that are required to go to the next state, e.g. in a setup the software will check if the system needs to be calibrated before starting and will conduct the calibration before the system is changed to the standby state. Depending on the module each state will have its own implementation depending on the specific hardware. In some cases a setup or start can be aborted. However, this will result in a change to the opposite transitional state, which will check if any actions already conducted in the original state needs to be stopped or shut down. In Figure 7.3 this state template is shown.

The definition of the states are as follows:

- Safe - The module is powered down and is inactive.

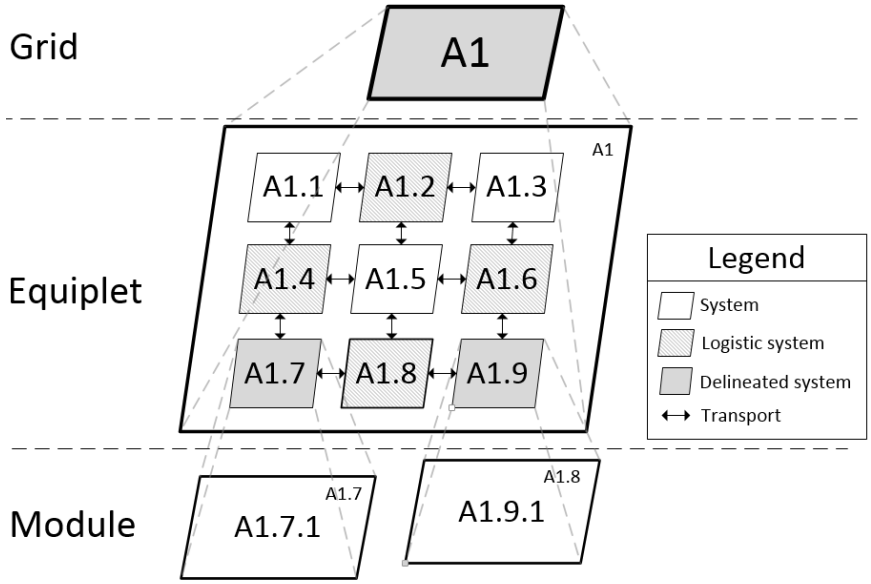


Figure 7.2: Grid system overview - GEM architecture - Grid, Equiplet, Module.

- Setup - Will conduct a start-up sequence, including configuration, calibration and possibly a self-test.
- Standby - The system is ready, but currently inactive. The actuators are powered and ready to start, i.e. the system will check for commands from an operator, or in Grid Manufacturing, a product agent.
- Start - The system has received a command and will start its operation.
- Normal - The system is active and running.
- Stop - Actuators are brought to their starting positions and the system is brought to standby mode.
- Shutdown - Actuators are powered down and the system will become unavailable for operation.

7.3.8 Modes

The module states provide a clear overview and are basically state machines that represent basic hardware operation. With classic line manufacturing these states could easily be controlled in a centralised manner. An operator would

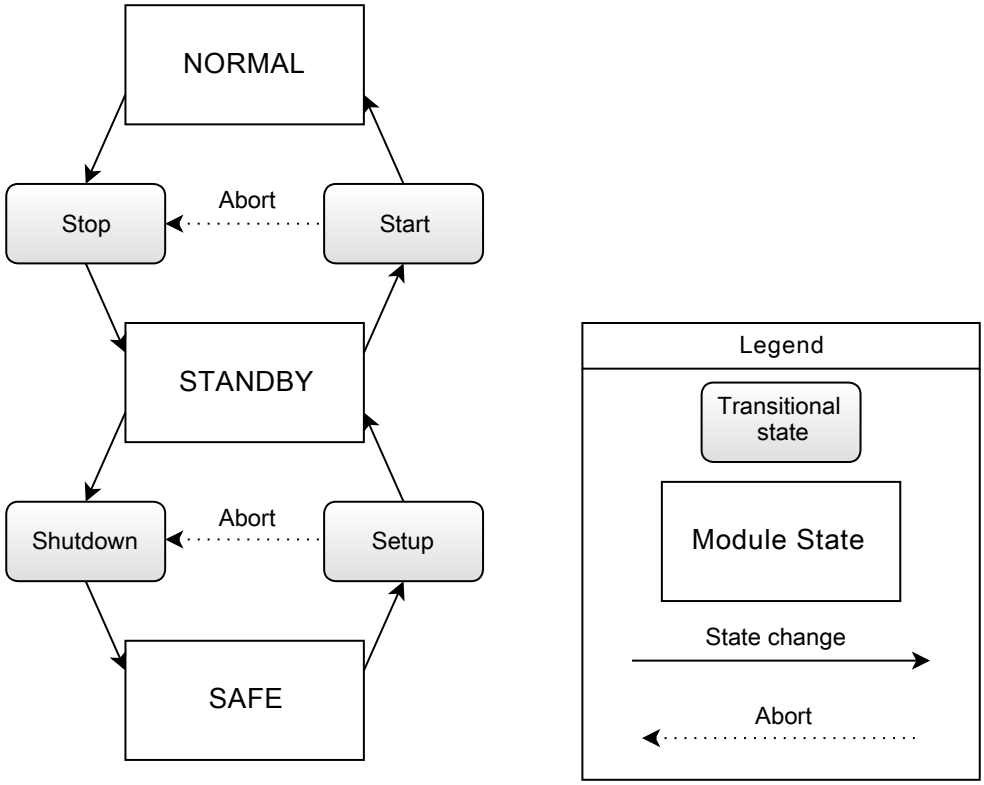


Figure 7.3: Module States (MOST) hardware representation.

give the command to go to a setup state and all lower systems in the hierarchy would follow; when all systems report standby, the start command would be given and all systems would start and go to normal, if necessary in a synchronised manner. However, in grid manufacturing this is not the case. Systems can start and stop not based on a centralised command with a push of a button, but because a product agent requires a service from a specific equiplet agent. This also means that a start can happen unexpectedly and the machine could start moving without warning, which could be a cause for danger. In this and other cases it is important to expand on the standard machine states and create specific behaviour. This behaviour can either be initiated as an automatic response in cases like an error, or as a specific extension on normal behaviour, like debug or service behaviour, which can be a response to a specific command from an operator. We call this extension on the standard states a mode, see Figure 7.4; here several modes are shown. In a mode the states can be redefined or extended, in some cases the mode can be extended with extra possibilities or instructions, like with a pause command in a debug

mode (which is not shown in the figure, since it shows the same states as the normal mode).

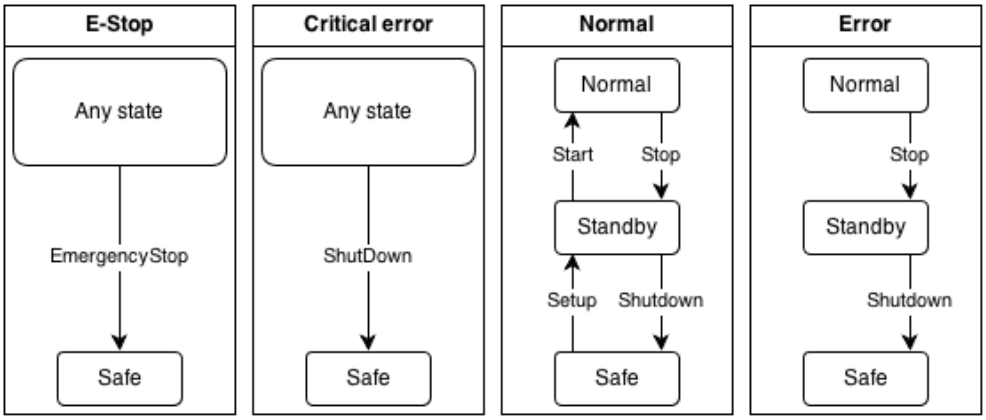


Figure 7.4: Modes provide a practical implementation to overwrite or redefine specific behaviour required for some situations like errors.

In some cases the modes will also change the possible state or state transitions. For example, in case of a critical error the system will be forced into an immediate shut down to power down all actuators. It is impossible to set up the system after this happens until the problem has been fixed and the system has been switched back to the normal mode. While the emergency stop (E-stop) has the same behaviour as a critical error, the transitional state is redefined. An E-stop always triggers a, fully redundant, electrical system that decouples all power from the actuator. Hence, in the case of an E-stop the modules do not have to be stopped. In this case only the software systems will remain online to perform damage control with the passive hardware (sensors) and software systems that are still active, e.g. a product agent can likely still be informed that its schedule will likely be delayed, which might trigger the product agent to reschedule at another equiplet that can provide the same service. In case of a normal (non-critical) error, the system will try to continue operation and come to a standby. In the error mode the standby can be interpreted as a stop, since a start will be impossible until the error is resolved. In manufacturing this might happen when a dispenser is empty and needs to be manually refilled. Since in error mode the system cannot start unexpectedly the system can safely be approached. However, actuators might be powered if they are still in standby state. If necessary the system can be brought to a safe state.

7.3.9 States Throughout the Grid

Module states and the modes provide a basis for hardware state representation and therefore safety in autonomous systems. Moreover, the module states, in combination with metadata, can provide a better ability to provide information about the grid. In this section we look at a part of the current grid implementation to discuss which states are required to oversee the states in a grid.

Implementation follows the GEM architecture as shown in Figure 7.5. The ROS nodes are used for direct hardware control and the agents provide decision abilities and diverse functionalities on the Equiplet and MAS layer. Take note that a grid can consist of many equiplets, which each have their own set of ROS Module nodes.

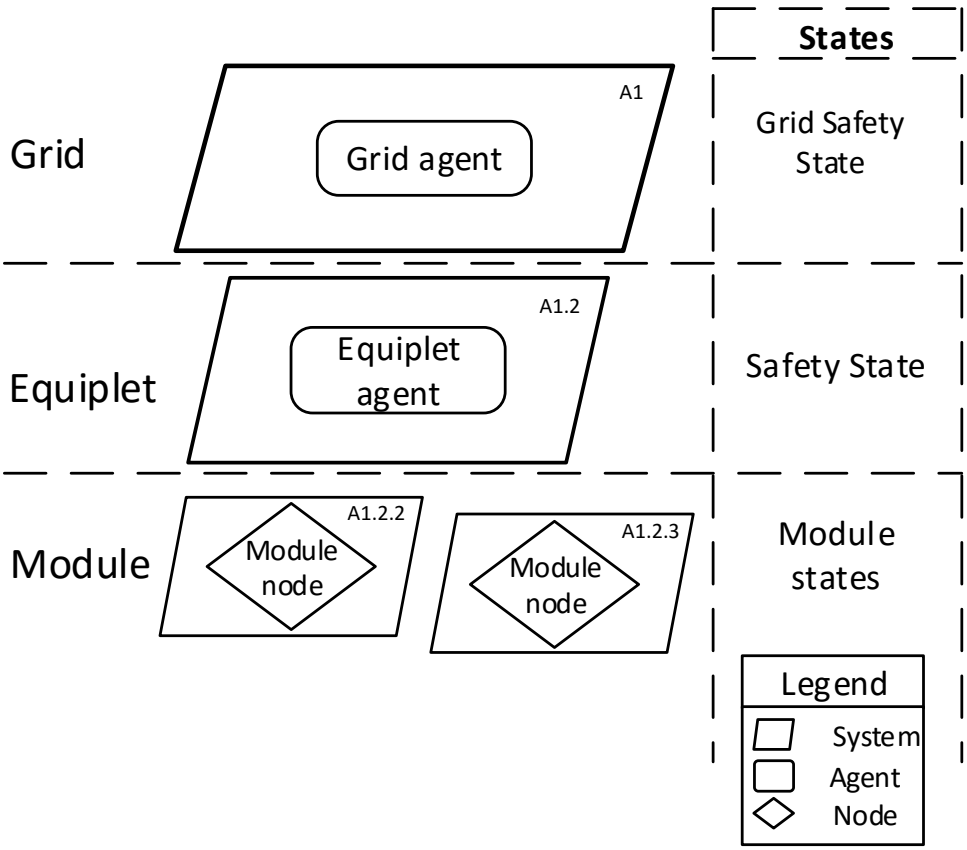


Figure 7.5: States concerning safety in the GEM architecture.

- The module nodes directly control the hardware modules and lower devices. On this level the module states are implemented and available for the equiplet agent to use.
- The equiplet agent directly controls all hardware modules in an equiplet, it has more knowledge of the system and can translate (abstract) services to instructions specifically for the configured modules.
- The grid agent has no control purpose, but only provides the combined status of all equiplets to show if all systems are safe.

Figure 7.5 also shows three states in the grid, which are specifically:

- Module state - A direct representation of the state where the module hardware is in.
- Safety state - The overall state of all actors. Actors are modules that can either move or have any other active parts, e.g. a heater. It uses the knowledge of the hardware agent to determine which modules are actors and gives the highest state of all actor module states to show if the entire equiplet is inactive.
- Grid (safety) state - The highest state of any equiplet in the grid, such that the grid is only determined safe if all systems in the grid are.

The module, safety, and grid (safety) state will provide a complete overview of activities within the grid.

Grid Behaviour and Opportunities

While systems act autonomously, the states actually provide insight throughout the grid to what is happening. Statuses are updated in real-time and therefore autonomous agents can react to events that happen in another layer. This provides the capability for a product to reschedule its steps at another equiplet if its deadline could become unreachable through an error at a scheduled equiplet. Also a service agent could pro-actively inform all scheduled products that operations might be delayed if a critical error occurred and the hardware unexpectedly goes to a safe state.

7.3.10 Discussion

A state system has become more complex due to the use of autonomous and reconfigurable systems. This is due to the fact that hierarchical systems are easier to define and the dependencies between classical hierarchical systems also mean that one system directly influences the state of another. Hence, due

to the more complex organisation of autonomous systems it is required to look at distributed systems instead of a classic SCADA system.

The grid manufacturing approach we proposed provides a definition and standardisation of states throughout the grid. This simplifies safety aspects and improves predictable behaviour when reconfiguring systems and during error situations. Grid manufacturing also increases flexibility by providing a heterarchical software architecture where product and equiplet agents negotiate directly to create new products. Because logistic systems, like conveyor belts, are also implemented as (logistic) equiplets, the entire *software* process, seen from a product's perspective, can be seen as a heterarchical system in the classifications as defined by (Trentesaux, 2009). However, looking deeper into the system and considering the perspective from a hardware control gives a different view. Hardware will always be hierarchical in nature, i.e. any hardware system will always have a master-slave relation since a (possibly) electric signal will always be a trigger for some device to start working. Seen from this perspective, equiplets themselves are set up hierarchically, which arguably makes a grid a class 2 semi-hierarchical system. The same can be said for logistics, which creates a dependency between equiplets. In the future these dependencies are planned to be optimised with the use of logistic agents that monitor the states and load on the logistic equiplets to possibly resolve these problems. While the details on logistics fall out of the scope of this Chapter, they are mentioned because a logistics agent on the grid size will likely influence systems on the equiplet level, making the system a class 2, semi-hierarchical control.

A standardised approach was discussed that provides a direct representation of the hardware, but also how this information is distributed throughout the system. This makes it possible for other autonomous systems to react and take action accordingly. Finally, handling errors and predictable behaviour are treated using the modes system. Modes provide a practical, i.e. pragmatic, implementation to overwrite or redefine behaviour required for specific situations. A specific mode can block unexpected actions of an equiplet and provide that extra flexibility required to handle practical problems like error handling, or specific service capabilities. A new aspect of this work is also the split in 'safety' state and the module 'activity' state. Since these are split, sensors can still be used without safety problems, increasing the flexibility of the system.

7.3.11 Machine State Conclusions

This section was given its own three Research Sub-Questions:

RQ4d. How can you better define the states that represent the autonomous systems? Equiplets are not just on when the start button is pushed, or off

when the job is completed. When an equiplet agent is operating normally, jobs are started dynamically in negotiation between the product and the equiplet. To be able to handle this, a classic state system was designed that gives clear definitions of each module state. On top of that, a safety state was created that can force the equiplet with its hardware modules into a safe state. This way both flexibility and defined system behaviour are known.

RQ4e. How do you handle safety aspects in these systems? The safety state on the equiplet level can enforce a state within all actor nodes within an equiplet, which gives a generic sense of safety, since it gives an expectation of what the system is (not) able to do. However, another aspect of safety is the physical safety, which will be discussed in the next section.

RQ4f. How can error behaviour be standardised when using reconfigurable systems? Modes were introduced to deal with non-standard situations. An error forces the states in another mode that redefines their behaviour. As such, equiplet control systems will be forced into a stop and shutdown mode, depending on the type of error. An error state can also be induced if an equiplet takes too long to finish a transitional state, since this likely indicates a problem.

In general this section on software control shows the organisation with a definition of states, the mode system, and state representation for Grid Manufacturing.

7.4 Physical Safety

This section focuses on the second part of the System Behaviour research. Its primary focus is providing 'physical' safety in dynamic environments when using reconfigurable manufacturing machines. The last section shows how software status is used to provide insight into whether the system is active or can be approached safely. Basically it increases safety by giving more insight in the status. However, this does not provide insight if a system is 'safe' when it is active, e.g. if it moves without hitting any objects or stresses certain parts, e.g. joints to positions that might break them. In this sense this safety can be seen as the Self-Protecting category from Autonomic ComputingComputing (2003). Equiplets are meant to identify possible threats, even though they are reconfigurable and work with a range of a priori unknown products.

This so-called 'physical' safety while the system is operating is an aspect that is discussed in this section. Parts of this work origins from an unpublished technical document (Bakker and Telgen, 2015).

7.4.1 Problem Description

While the state and control of the reconfigurable manufacturing was handled in the last section, another problem is still not solved. This is the physical safety aspect. Possibilities for collisions within the used systems are high. Both products and hardware systems are not known *a priori*. Hence, it is essential for safety reasons to identify if the machine can act, i.e. move, safely. While the MAST system gives permission to a system to perform some action this does not give any guarantee that the actions themselves are safe. Since hardware can be reconfigured and products can be of any kind it is important to create a system that checks if any physical movement could be a safety risk.

7.4.2 Proposal

Safety concerns with robots occur when they are performing a task, such as grabbing an object. The robot might damage itself or its surroundings while performing a task due to a (control) software error or incorrect sensor data. This behaviour might damage the robot or its surrounding. A way to make sure the robot will not damage anything, is to actually let it perform the task and check the end results. A simulation makes it possible to do this safely. By simulating the robot and the environment, the robot should behave exactly as it would do in the real world. For this to work, it is essential that the simulation is an exact, or a very close, replica of the real world. Differences between the two might cause the robot to behave differently.

An important aspect of the simulation is that it is able to run within a real, active manufacturing grid. This means that the simulation should be integrated within the REXOS architecture, being able to run side by side with the actual systems.

Scope

Only physical safety concerns that do not involve external influences are considered. In the simulation, the manufacturing systems and the products are seen as a closed environment, i.e. the scope of the simulation is limited to systems that are controlled by the GRID software. That is to say, human operators are not within scope; it is expected that they are aware of the hardware action range and monitor the MAST state which was discussed in the last section to determine if a machine can be approached.

7.4.3 Sub Research Questions

The main Research Question for this section is: To what level can simulation & modelling determine the safety of the system behaviour of a reconfigurable

production machine that works in a dynamic environment?

This question can be split into four sub-questions:

RQ4g What are safety concerns, and how can simulation address them?

RQ4h How can new models be added dynamically?

RQ4i How can the safety checks be integrated in the simulation?

RQ4j How can the simulation be integrated in the current REXOS system?

7.4.4 RQ4g - Safety Concerns

It is important to determine the safety concerns related to production machines to be able to set the requirements of the simulation. With the requirements known, a suitable simulator for the simulation can be found.

The following five safety concerns were identified:

- Objects colliding with too much force.
- Objects being exposed to too much stress.
- Objects becoming too hot or too cold.
- Joints going beyond their limits.
- Objects being exposed to air for too long.

The answers to the other research sub-questions of this section depends on the used simulation. Hence, we first discuss the possible simulation engines and choose one to develop and test a simulation that can be used for equiplets.

7.4.5 Simulation Engine

Many simulations only simulate logistic actions and are therefore usually in 2D, because the third dimension is not relevant, e.g. if the simulation consists of robots driving on the floor of a room, the third dimension might not matter because the robots will always remain on the floor. For those applications simulators that only support 2D simulation are commonly used. However, because the robots used for REXOS have robotic arms that can move in more than 2 dimensions, the simulator must support 3D simulation.

The following three considerations have been taken into account:

- Does it support the simulation of fluids such as water? The robots in REXOS are not made of fluids nor do they encounter fluids. Scenarios

with not fully dried fluids (such as adhesive or solder) which could deform when not handled with proper care are plausible. However, these scenarios do not require full fluid simulation. If necessary, limiting the acceleration rate should also suffice to lower the risk of not fully dried fluids.

- Does it support elastic materials? These materials bend when sufficient force is applied. This is not relevant for the robots used for REXOS as they are all made of rigid materials. Scenarios with elastic materials which could deform when not handled with proper care are also plausible, but also do not require full simulation as this could also be mitigated by lowering the acceleration speed.

- Does it support physics? This is very important, as some robotic arms configurations developed for equiplets are complex and use multiple motors rotating in conjunction with each other. To properly simulate these arms, a simulator that supports physics is required as the final position of the arm is determined by the forces applied by the motors. A physics engine also allows measurements of forces applied to components of the robots or product. This allows the simulation to determine if certain forces exceed the maxima of the components.

These considerations have led to three possible simulation engine candidates:

1. *Gazebosim*[1] is an open-source simulator using OGRE3D[2] for its rendering. It supports physics via the Open Dynamics Engine (ODE)[3] physics engine, but also supports other physics engines. Gazebosim only supports rigid bodies and has experimental implementation of fluid simulation using particles. An interface between Gazebosim and ROS is available. Gazebosim is licensed under Apache 2.0.

2. *Webots* is a commercial simulator using the ODE physics engine. It offers various interfaces such as a Matlab interface, an URBI interface and also a ROS interface. The simulation is feature-rich, as it enables users to monitor various sensors (such as pressure sensors) visually. The simulator requires a license. To check the validity of the license, Webots requires a constant internet connection.

3. *USARSim* is a simulator based on the Unreal game engine. The rendering and physics engines both use the game engine for these tasks.

[1] http://gazebosim.org/ - last accessed 23-03-2016
[2] http://www.ogre3d.org/ - last accessed 23-03-2016
[3] http://www.ode.org/ - last accessed 23-03-2016

Because the internal API of the unreal engine is commercial propriety software, a reverse engineered API has been developed. Since the USARSim uses the Unreal engine, environments can be constructed with the Unreal Editor.

REXOS runs both in Java and C++. The C++ side of REXOS uses the ROS framework for inter-process communication. Having a ROS interface is therefore a pre. Gazebosim offers this interface, is open-source, and is free. Gazebo is also recommended by the ROS community for simulating 3D models. Therefore Gazebosim is considered to be the most suitable, and will be used as the simulation platform for equiplets.

Gazebo makes use of models, since these have different properties that are important for the physical safety, commonly the Simulation Description Format(SDF)[4] is used. The SDF format was specifically developed for Gazebo, with scientific robot research in mind, and describes objects and environments in an XML format.

This format uses some specific components that are important to be able to understand some of the answers to the research questions:

- Links - Describe the physical properties of one 'body', e.g. a wheel of a bicycle. A link may contain a whole number of elements, including:]
  - Collision: an element that encapsulates a geometry to detect if 'collisions' occur.
  - Visual: visualises a part of the link.
  - Inertial: describes the dynamic properties of a link, e.g. mass.
  - Sensor: collects data from a plug-in.
- Joints: describes the connection between two links.
- Plug-ins: a shared library from a third party that contains specific abilities to control a model.

Now that the simulation platform is established the following research questions can be answered.

7.4.6 RQ4h - Adding New Models to the Simulation

Because the production machines are reconfigurable, the simulation must support the addition of new hardware module models. This includes completely new models that have never been used before. Since in Grid Manufacturing

[4]http://www.sdformat.org/ - last accessed 23-03-2016

new products can also be introduced, it is also important to be able to add models of products and parts. All the data (including the safety aspects) of the module must be interpreted automatically and the simulation must adjust accordingly.

When possible, it will allow REXOS to add models dynamically through the use of ROS. Unfortunately the interface does not provide a method to attach a (sub)model to another model, this provides a challenge, since the reconfigurable aspect of equiplets implies that modules can either be connected to an equiplet or to another module. Since these modules each have their own module it is essential that to simulate them they should be able to be 'connected' in the simulation. If this is not possible the model added to the simulation will simply fall on the ground. However, there are several methods to remedy this problem:

- Define a static component in the model or attach the model to the world by determining fixed locations. Either method will ensure that the model will remain in its position. However, it will become impossible to move the model in any way. For some modules this will suffice, for others it will not. A gripper piece attached to a robotic arm, for example, will not work with this method as the gripper piece must match the movements of the robotic arm.

- Define a Simulation Description Format (SDF). An SDF with a gripper object could be used to attach two models to each other. To create a connection between two models, both must have a 'collision' element and one must have an SDF gripper object. In Gazebosim a *collision element* is a specific physical property that defines the shape of models for the physics engine. More specifically the 'collision element' encapsulates an object to perform collision checking. This might be confusing in this context, since the same word 'collision' can both be used for the 'collision' element as well as the event of a collision (two objects colliding). In this context, Collisions, i.e. multiple collision elements, can collide. If this happens, the collisions have contact. When the gripper object comes in contact with the collision element of the other model, it will automatically create a connection and thus attach the two models. This method is suitable for this problem, since it allows models to match the movements of their parent models. However, it is an expensive method as it requires additional collision elements to be defined. This has significant impact on the performance of the simulation as each collision element has to be matched with every other collision element to determine whether or not there is a legitimate contact between them. Another problem is that the two models do not really collide with each

other, they are only adjacent to each other. While this too should count as a contact, rounding errors in floating point data types might cause the physics engine to miscalculate and causing the gripper object not to connect with the other model.

- Generate an entirely new SDF file containing all the models connected to each other. An external program is used to retrieve all the SDF files from the individual models and generate a new SDF file. This new file will become very large and thus very hard to read and debug. Therefore this method is hard to maintain. A second problem is that the new SDF file will contain a model that replaces all the old models, meaning that every modification to the robot will result in having to replace the entire robot. This not only takes longer, but also resets the entire robot model. If a robot arm was locked in a certain position, a discrepancy between the software and the model will occur after the reset. The software will have to be restarted, told that the model has been reset or will have to recalibrate. Either solution will involve modifications and discrepancy between the behaviour for the simulated robot and the real robot (the hardware of the real robot does not suddenly reset to a different position).

- Write a custom plug-in that will dynamically connect the model with the other model specified in the SDF file. This will still require an external program to generate a new SDF file but it will only replace the parameters for the plug-in. Once the model is loaded into the simulation, the plug-in will start and immediately connect the model with the one specified in its parameters, by adding a joint to the simulation.

The last option is chosen, since it is the most flexible, the easiest to implement and has no major downsides.

7.4.7 RQ4i - Integrating Safety Checks

The simulation must guarantee safety. Hence, safety checks must be integrated. It is important to determine how a safety constraint can be specified in a way that the simulation can interpret. Also it must be determined how the method for reporting the results of a safety check back to the system can be performed, without requiring extensive modifications to the existing software.

All the data associated with the models including the SDF files, meshes, textures, and offsets must be properly managed so it can be inserted into the simulation when necessary. REXOS already uses a method for managing software components which is also suitable for managing the models. This method uses a relational database called the Knowledge Database (KDB) for

long term storage purposes. Figure 7.6 shows the entity relation diagram for the related part of the KDB.

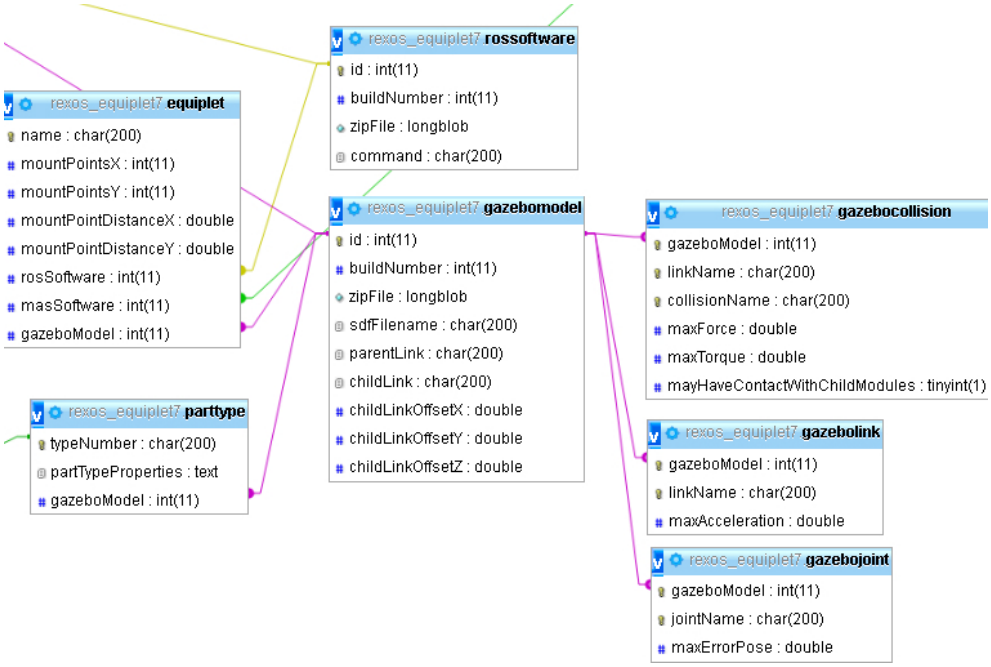


Figure 7.6: Cut-out of the entity relation diagram of the Knowledge Database.

In REXOS the software components are packed in ZIP files and extracted by a ROS node which will also start the new software component. When a module of the robot is added or removed, the KDB will be updated. The data of the new module is provided via a large Javascript Object Notation (JSON) file called the Static Settings file. JSON is a very flexible open-standard format. The data is de-serialised, i.e. taken out of its storage-specific format, and inserted into the KDB. When a module is removed, this process works in the exact opposite way: data is removed from the KDB and serialised to form a static settings file. The same process can be used for the models, this makes the REXOS system more consistent. The SDF files, meshes, and textures are packed in a ZIP file. The ZIP file is extracted by a ROS node called the model_spawner_node when the model is to be inserted into the simulation. The insertion and removal of modules works in the same manner. The models do not only contain SDF files, meshes, and textures but also constraints. These are stored in their own tables at the KDB.

Joint Limitations

Components are connected by joints (in Gazebosim these are called links). In context of the human body one could say that the shoulder is a joint, as it connects the arm to the rest of the body. The same applies to robotic arms and their models. Some joints can rotate freely while others have limited freedom. Trying to rotate joints beyond these limitations might damage them. Hence, the limitation of a joint is a (physical) safety concern. Joint limitations for robotic arms or other components are defined in the SDF files of the models. The simulation will prevent joints from rotating beyond these limitations by applying a very strong force on the pieces connected to the joint, attempting to rotate the joints back within limitations. This force will rotate the links connected to the joint so that they match the constraints. There are scenarios where two joints attached to the same link apply forces that counteract each other, causing the link to remain in the middle between the two joints. If this happens, it means that the physics engine is not able to resolve the model to a stable state, and will continue applying forces on the link.

The distance between the joint and the link is called the error pose, i.e. an erroneous position, and can be measured. See figure 7.7 for an example of a joint that is at the wrong position due to forces pushing it away from the joint. The error pose increases as the joint limitation is violated further. Because the error pose equals zero in a stable situation, it can be used as a method for determining whether or not the joint limitations are violated by providing a maximum allowed error pose. The error pose is measured and compared by a plug-in running in the simulation.

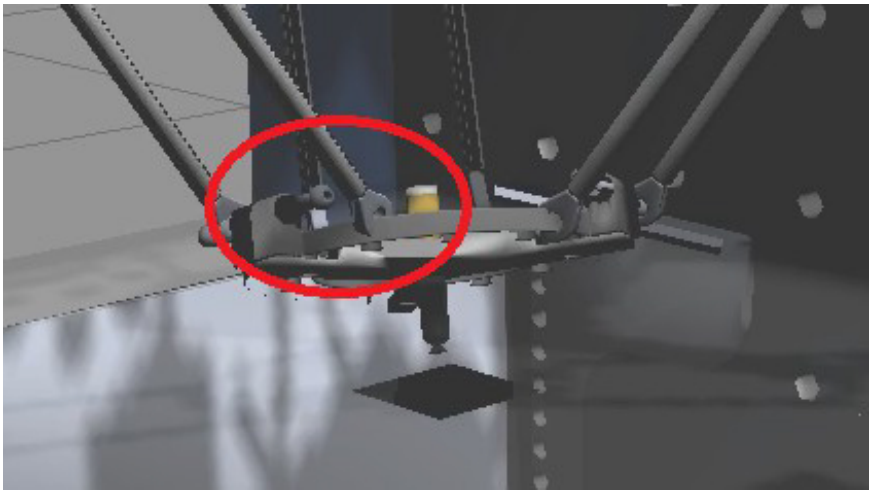


Figure 7.7: Limitations of the joint causes an error pose, where the joint is detached from the connection point.

Collision and Contact Force Limitations

Collisions between components result in large forces applied on these components. This could damage the components, requiring replacement and causing downtime. This is a safety concern and therefore it is important to prevent these violations from happening. This can be accomplished by simulating the robot and checking for any collisions.

Some materials can handle impact forces better than others, e.g. a wooden component may bounce, while a glass component will break. The simulation is able to determine the force and torque applied to the links as a result of the impact. By specifying the maximum acceptable impact force for each link, the simulation is able to determine whether or not the acceptable impact force limit is violated.

This is one of the most important aspects of the simulation, since accidental collisions between objects, like products and the equiplet, will need to be avoided. With the use of the 'collision elements' mentioned before, and the use of a plug-in that monitors the force of these collisions, it can be measured if an operation is safe to be run before it is performed with the real hardware.

Acceleration Limitations

Exposing components to high acceleration might cause them to deform or lose subcomponents. When they are sufficiently deformed or have lost important subcomponents, they need to be replaced. This causes downtime and is therefore one of the physical safety concerns. However, the simulation can determine the position of a component. By measuring the distance covered and the elapsed time, the velocity can be determined. By measuring the change in velocity over time, the acceleration can be calculated. Because components tend to shake violently when in contact with other components, the acceleration is calculated with 51 samples to smooth the acceleration. The shaking is caused by the way the physics engine handles contacts between objects. The acceleration is measured and compared with the limits by a plug-in running in the simulation.

7.4.8 Implementation

The Gazebo simulation seems able to provide the ability to use or develop the necessary plug-ins that will monitor all physical safety concerns, and has the correct capabilities to ensure more safety for the REXOS system. The models for the equiplets and modules were already available, since it has been implemented using the models that were used to design and manufacture the hardware for the equiplets themselves.

Figure 7.8 shows the GUI of the simulation. Both products and equiplets with different modules can be configured using a command line interface.

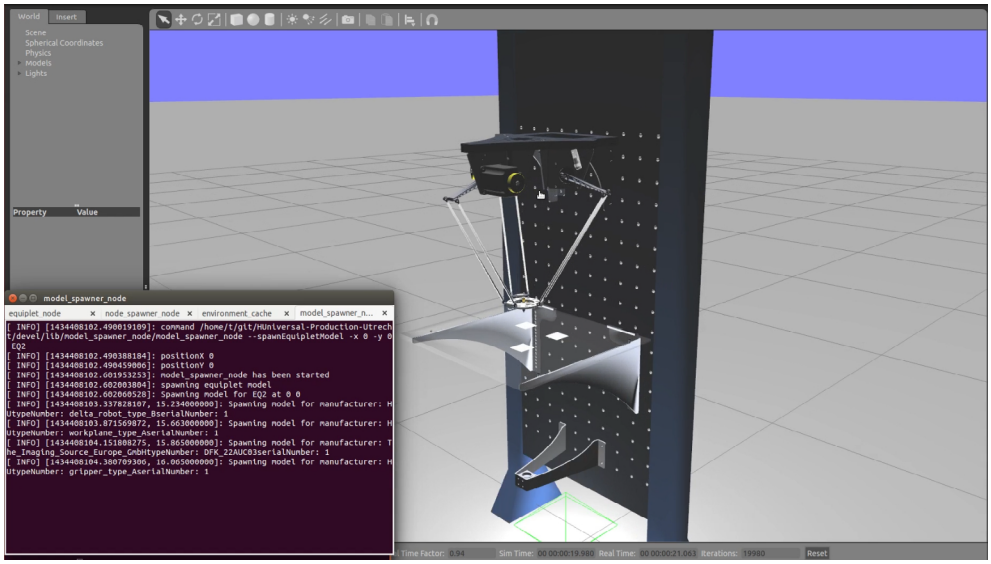


Figure 7.8: The Gazebo simulation can create equiplets in any configuration.

In reality the simulation can be used to test all actions that an equiplet can take, the plug-ins will raise alarm when one of the safety concerns violates one of the given restraints. However, for the system to be effective it should be integrated within the REXOS architecture, this will allow to test real cases within identical code in a live system.

7.4.9 RQ4j - Simulation Integration

The simulation will be used by the REXOS system and therefore has to be integrated. An integration method with a clear interface and without excessive code duplication is essential. Therefore it is important to determine what the best places are to integrate the simulation with REXOS.

In REXOS there are product agents (PA) and equiplet agents (EA). Product agents represent a product and the first goal of the product agent is to get itself manufactured. Therefore the product sends product steps to the equiplet agents. A product step describes a desired output of a task, e.g. place part A 2 millimetres above part B. It does not concern itself with the actual task required to make this happen. The equiplet agents will receive the product steps. The equiplet agents represents the robot capable of performing tasks. As described in Chapter 5 it will use its Hardware Abstraction Layer (HAL) to generate a task to reach the desired output (in REXOS this is called

translation). A generated task is composed of multiple hardware steps, each describing an action that needs to be performed by one of the equiplets actuators. The hardware steps are then sent to ROS, which will forward them to the correct actuator (module). The module software interpret the hardware step and control the hardware accordingly.

To guarantee physical safety it is required to test each step before it is performed with the real hardware. However, this can only be done with the translated hardware steps that have been translated by the HAL for this specific hardware. Hence, the testing within the simulation will need to be performed between the translation of the product step and ROS performing the hardware steps. When testing, the hardware steps will go through the exact same procedure as when performing the real robot performs them.

For the integration two integration points are chosen; a high integration point at the MAS level, where the steps are sent to ROS for testing purposes and a low integration point where the module control the hardware in the simulation. Figure 7.9 shows the process that is used when the simulation is active.

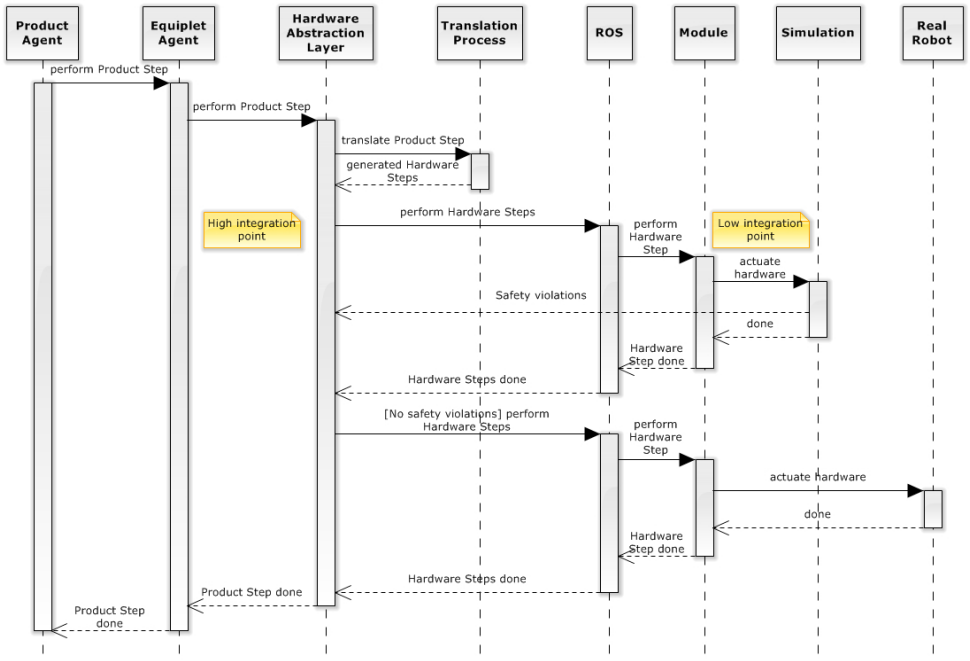


Figure 7.9: Sequence of translation, testing, and performing a product step.

The high integration point will capture messages to the real hardware and will use the standard code to translate and run them through the simulation.

When no safety violations occur, it will rerun the commands through the normal 'live' hardware systems that will actually perform them. However, since the simulation also uses the standard code that would normally directly control the hardware, it is also required to create a 'low' integration point that takes over the low-level interfaces to the hardware. Both will be explained in more detail.

Low Integration Point

Because the robot in the simulation should behave exactly as the real robot, the integration that recycles the most software is the most preferable, as it has the highest chances of correctly matching the behaviour of the real robot. This way, existing software errors should occur in both cases. This means that the integration of the simulation should occur at the lowest level possible. In this case this is the interface to the hardware. By providing a generic interface to the hardware and implementing one for the simulation and one for the actual hardware, all the higher levels (including various calculations) do not require any modification and can be used either for the simulation or the real robot. For example, the input-output controller class, used for communicating with various hardware such as sensors and valves, has an interface that communicates with the higher classes. The input-output controller has two real implementations (both using an industrial network standard, which is called *Modbus*) and one simulated implementation (using ROS services). This is shown in Figure 7.10.

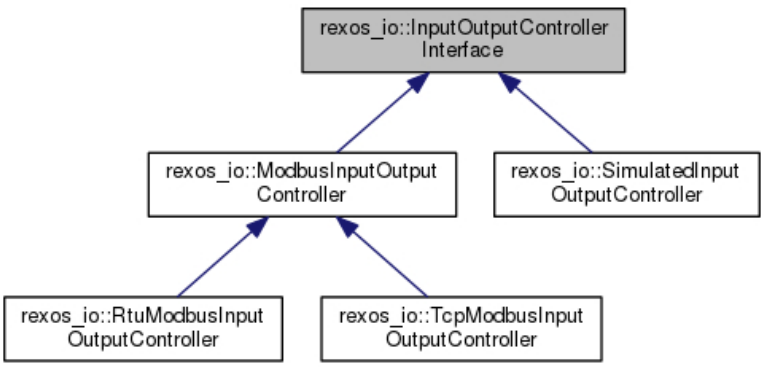


Figure 7.10: Class diagram for the input-output controller that can switch between simulated or real IO.

High Integration Point

The best high-level integration point is mostly determined by the same factors as the low-level integration point. However, the simulation should not be exposed to the equiplet agents or other components in MAS. Yet it is important to recycle as much software as possible as this increases the changes of the simulation matching the behaviour of the real robot. Hence, the simulation test procedure is integrated on the hardware abstraction layer level. This will enable to simulation of an entire equiplet, including its specific configuration with all modules and other hardware. To be able to test all functionality the simulated equiplet has its own ROS nodes and its own hardware abstraction layer, which are identical to the real software that is used for the actual equiplet.

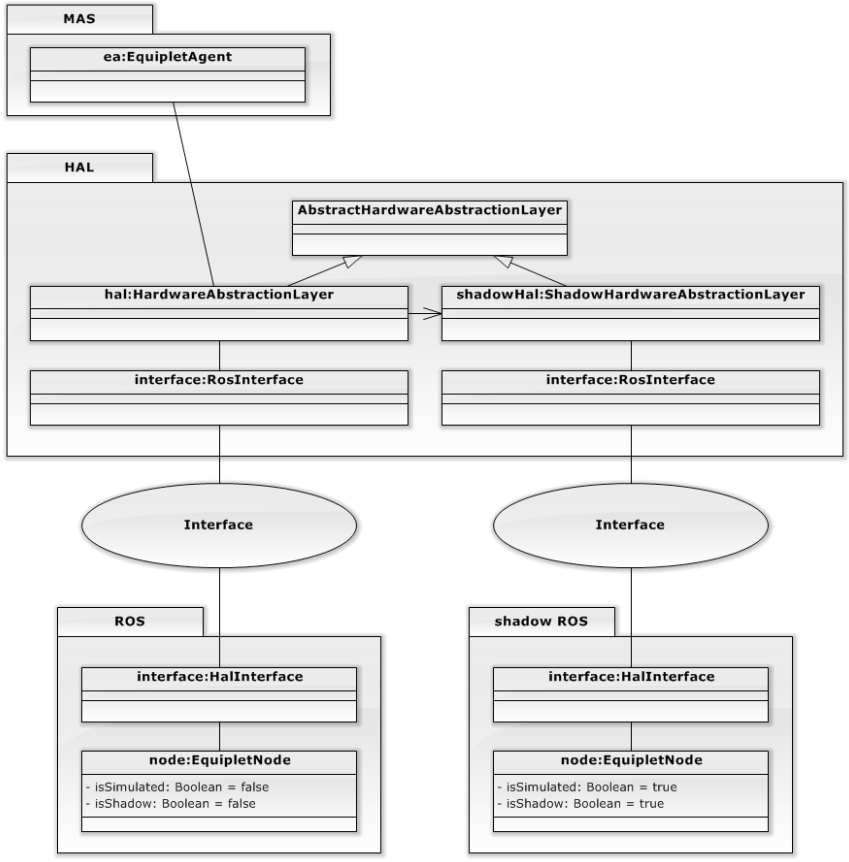


Figure 7.11: The object diagram of the REXOS infrastructure with on the left the real robot and on the right the simulated robot.

7.4.10 Simulation Opportunities

The original purpose for the simulation was to be able to test various physical safety aspects. However, by creating a simulation many other opportunities have arisen.

With the simulation any equiplet configuration can be created and tested with the actual software that would also run a 'real' equiplet. The simulation is able to add modules, products or parts at any time. When adding objects the user interface will inquire for a number of parameters, e.g. where to place an object within the simulation environment. Figure 7.12 shows the simulation, with the command line interface that starts new systems within the equiplet simulator.

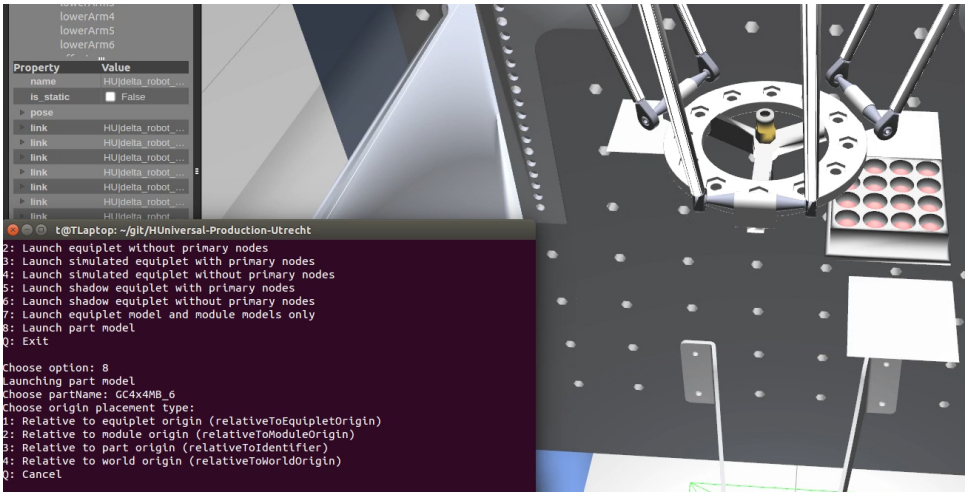


Figure 7.12: The running simulation with the Command Line interface that could be used to add any equiplet or object 'on the fly'.

The simulation can be used to run fully simulated equiplets or even a full grid with multiple equiplets. However, since the simulated robot can match the exact behaviour of a real equiplet, it can be used to verify the exact steps that an equiplet will perform. This is exactly what is done for the safety concerns. When a real product step will be performed that could be a risk, e.g. if a new product or configuration is active for the first time, it could be tested with the simulation before the actual step is performed in reality. In this case, the simulated equiplet can be seen as the (Cyber-Physical) *shadow* of an equiplet. Only in this case, the shadow acts 'before' the actual system does. Hence, simulated robots used for testing purposes are actually called 'shadows'. The shadow is always simulated. However, the 'real' robot could also be (fully) simulated, since this requires no modifications to the system.

Simulating 'real' robots enables testing of environments with multiple robots without having to acquire the hardware and thus saving costs.

This results in three types of robots:

- Real robots: These are real physical robots, existing in hardware. The ROS nodes communicate with real hardware.
- Simulated robots: These are simulated robots, existing fully in the simulation. They can be used for running a simulated grid without requiring a physical counterpart.
- Shadow robots: These are simulated robots, existing in the simulation and are used for running tests for another (either physical real or simulated) robot. Shadow robots cannot work standalone, i.e. they always require a counterpart.

Since all the robot types can be combined it is possible to test a complete production process with both real and simulated robots. This could be used for a variety of possibilities for both testing and optimisation opportunities.

7.4.11 Physical Safety Conclusions

The following research sub-questions have been discussed in this section: *RQ4g - What are safety concerns and how can simulation address these?* Safety concerns are conditions/problems that will result in damaged hardware. Simulation can address safety concerns as it provides an environment where the robot can safely attempt to perform its task. If the simulation is an exact or very close replica of the real world, the behaviour of the robot should be the same.

To properly simulate the robots used for REXOS, a simulator with a physics engine is required. Fluid simulation and elastic materials are not required. Gazebosim has been chosen as the most suitable simulator as it is open-source, free and offers an interface to ROS. The Gazebosim has been created with a number of plug-ins to handle specific aspects that are important for use with equiplets.

RQ4h - How can new models be added dynamically? Models can be added to the simulation through the ROS interface. However, this interface does not provide a method to attach models to each other, which is required to be able to dynamically create different configuration of modules for equiplets. Hence, a method has been created to attach models to each other through a custom written plug-in, which will add a so called 'joint' to the simulation.

Because models can be added and deleted, it is important to properly manage the data associated with them. The already existing infrastructure

in REXOS is suitable to store and copy models. Using the infrastructure also simplifies the REXOS system and makes it more consistent. The models are extracted by a specific spawner node in ROS.

RQ4i - How can the safety checks be integrated in the simulation? The three safety concerns currently supported in the simulation are the joint constraints, the maximal force and torque acceptable for a collision with another component, and the maximal acceleration rate that a component can handle. These safety concerns are all monitored by plug-ins specifically designed to monitor these risks and alert the REXOS system if the step is not safe to be performed.

RQ4j - How can the simulation be integrated in the system? REXOS translates a product step into hardware steps. Testing of these hardware steps must happen after the translation and before the actual robot performing these. To integrate the simulation into REXOS two integration points must be addressed: The high integration point that receives the messages from the MAS layer and the low integration point that handles the interface to the hardware.

Because the behaviour of the robot simulation should match the behaviour of the real robot, it should also reuse as much code as possible. Therefore the low integration point shall be on the interface to the hardware. By providing a generic interface to the higher classes, switching between real hardware and simulated hardware is very easy and does not require any changes to the higher classes.

The high integration point is at HAL level. This reuses most software, while not exposing the simulation to the equiplet agent. Because the simulation simulates an entire robot, the software should also represent the entire robot. Hence, the simulated (or shadow robot) robot runs its own complete HAL and ROS software.

Simulated robots can be used as a mirror/shadow of a real robot or as a standalone.

The performance of the HAL-ROS interface has become very important, due to the simulation requiring more commands over the ROS-HAL interface. A good performance significantly increases the number of robots that can run on a single computer. However, this aspect was covered in Chapter 6.

The final conclusion for the physical safety section is the answer to the main research question of this section: 'To what level can simulation and modelling determine the safety of the system behaviour of a reconfigurable production machine that works in a dynamic environment?', together with the research goal to reduce the change of safety concerns being violated, which could result in damaged hardware. The goal was achieved by the simulation, which greatly reduces a number of physical safety concerns by being able to 'mirror or shadow' a real equiplet and checking the actions that it will take. This tests both the software as well as the physical safety concerns

that were discussed in this section. To answer the main research question: The simulation can greatly reduce the risks that have been discussed in this section, which includes the use of dynamic products and the reconfigurable aspect of the equiplets. However, the simulation is limited to aspects that are known in its model. Therefore unknown disturbances, such as an operator being in the vicinity, are not taken into account by the simulation.

7.4.12 Future Work

Based on the findings and results of this research, the following five recommendations are made for future work:

- Add additional models - To fully integrate the simulation into REXOS, additional models for all modules and parts must be provided. These models should have all their specific safety concerns specified. Only a limited amount of parts and modules have been modelled at this time.

- Add additional safety checks - Develop additional safety checks for the simulation. This reduces the chances of the simulation not detecting a problem. A possible additional safety check could for example be the gripper overheating.

- Determine the duration of product steps - Develop a method to calculate the duration of product steps using the simulation. This should improve the estimated duration and allow the equiplet agent to use tighter schedules and thus allow for higher productivity.

- Research quality control - Do additional research to use the simulation as a quality control/quality check method by analysing the result of the simulated steps. The behaviour of simulated robots matches the behaviour of real robots, the quality of the performed tasks should be about the same. The simulation also provides an API for determining the location of components. By comparing the end result in the simulation with the criteria specified in the product step, the quality can be determined.

- Research large scale system behaviour - Use the simulation for simulating multiple robots and the interaction they have with each other. Transport of parts should also be included. This allows testing the system behaviour of the entire REXOS system (the grid).

7.5 Discussion

The System Behaviour section focuses on problems that are specific for the environment in which a grid operates, i.e. dynamic use of reconfigurable machines and products. However, generic safety and behaviour aspects that are valid for any machine are considered to be out of scope. This is also why generic machine safety guidelines are largely ignored in this chapter. However, generic safety machine regulations are upheld in the actual implementation.

The simulation does handle a number of safety concerns, including the risk of collisions, it does exclude aspects that are not inside the model, e.g. human operators. However, the MAST system provide a generic basis to show if a system can be approached by providing a state. Since a grid will work in a professional environment it can be expected that all personnel are aware of the state and its meanings and can therefore safely consider when an equiplet can be approached safely.

From a machine state perspective it is challenging to define the REXOS system. In contract to classic manufacturing, all systems work autonomously and as such act, i.e. could change states, 'on demand' of the product agent. Hence, while Grid Manufacturing is in essence not a hierarchical system that is controlled from a central point, it is essential to be able to use hierarchical means to enforce a safe state when deemed necessary.

7.6 Future Opportunities

The most interesting aspect of future opportunities on the long term is to create a 'complete' Cyber-Physical System where all aspects of REXOS are both available in real-time in the virtual, and the physical world. If sensors will be added this could also increase the safety by adding unknown dynamic factors, e.g. human operators, into the simulation. Other opportunities in the short term could be:

1. Product step timing / Improved logistics
2. Quality control
3. Capability check

By performing a step in a simulation with the current configuration it would be easy to calculate the precise time that it would take to perform a product step. That will create the possibility to precisely predict how long certain product steps will take. This information can be used to improve the logistics and scheduling throughout the grid. Workloads could also be balanced on basis of this work.

Another aspect is also that of quality control. Since similar products can be produced by different hardware it is important to check what the quality of the performed steps were after they were performed. By performing this step in the simulation it would be possible to test if certain configurations perform better, e.g. place a part more precise. Theoretical consistency could also be checked this way.

Another aspect is to check if the performed steps really fall within the capabilities of a new configuration. New hardware and products could be tested with different configurations and see if the results fall within the correct capability and if this will still produce quality within the requirements.

7.7 Conclusion

This chapter has described a number of problems and risks that have been introduced due to the reconfigurable and dynamic characteristics of the grid manufacturing paradigm. Some include how a system behaves, which is described in the MAST section. The MAST section works with standardised states and modes that describe when a system is active or safe to approach. The other aspect that has been investigated is that of the physical dangers, which has been handled by creating a simulation that checks a number of risks before an action is performed.

To specifically answer the original research questions: *RQ4a - How can the status be clearly identified and shown when using a diversity of reconfigurable modules?* The MAST system describes that modules are aware of its capabilities and as such know if they have an actuator (that could be potentially dangerous) or only passive (sensor) systems. Based on this knowledge it is possible to be in a safe state, while some modules are still operational. The MAST system also gives an overview of all systems on any level within the GEM architecture.

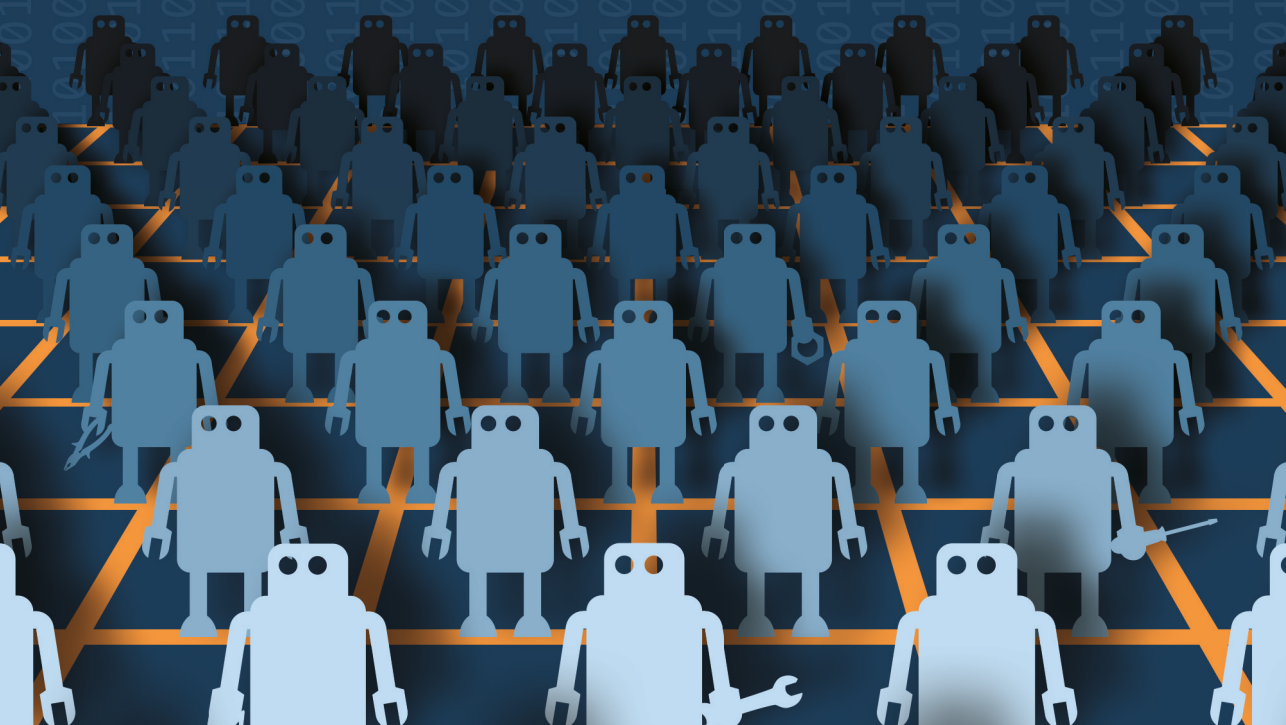
RQ4b - Can a classic state machine that is used for classic machines be adapted for reconfigurable machines that act in a dynamic environment? The MAST state system within the GEM architecture provides a clear hierarchy between modules and the overall control system. While equiplets and products behave heterarchically, the states within an equiplet are strictly hierarchically implemented and enforced so that no actuator can be used when an equiplet should be in a safe state.

RQ4c - How can we guarantee physical safeties when reconfiguring modules with different physical aspects? A simulation can 'shadow' a real equiplet. The simulation runs the same software and uses the real actual parameters to mimic the actions before the real actions are performed by the actual equiplet. Most aspects that are part of the models can be checked, including the limitation

of joints and collisions that occur. This greatly reduces the risks that occur when dynamically handling a diversity of products with different hardware configurations. However, it does not guarantee safety of use with objects, e.g. operators, that are not part of the Cyber-Physical model. However, these should trust on the MAST system to know which actions are permitted during what state.

Both MAST state system using modes and the MAST state together with the implemented simulation greatly improves the knowledge of the system and its behaviour. The simulation also provides a safe way to analyse the performance of a configuration or a new product that is introduced in the grid. Especially the possibility to run the equiplet as a 'shadow' to the real hardware is a powerful tool to dynamically test new situations and/or configurations. Hence, safety risks that are introduced by the reconfigurable and dynamic aspects of the Grid Manufacturing paradigm are greatly reduced by these systems.

08



Validation and Utilisation

“An expert is a person who has made all the mistakes that can be made in a very narrow field.”

– Niels Bohr

Validation and Utilisation

The last chapters have explored the technical aspects of the Grid Manufacturing paradigm. The next step is to see how grid manufacturing could be utilised in new ways and investigate the impact it can have on a logistic level. Note that the goal of this chapter is not to prove overall efficiency or optimise Grid Manufacturing systems. However, the chapter will show new ways to utilise the capabilities of a grid and explore a small amount of cases that are meant as a proof of concept. Also, it will discuss the impact of reconfiguration and the ability to automate manufacturing of products that were still undefined when the equiplets of the grid was taken into operation. Also, it will discuss ways to mitigate the challenges that these unknown factors introduce.

Parts of this chapter have been published at the 24th and the 25th International Conference on Flexible Automation and Intelligent Manufacturing (FAIM2014 & FAIM2015) (Telgen et al., 2014, 2015a) and have been used for and from the master thesis of (van den Brink, 2015), which was performed and supervised in concurrence with this research.

8.1 Additional Research Questions

The main Research Question for this chapter is: RQ5 - What is the best way to utilise the possibilities of an agent-based manufacturing grid and therewith validate its efficiency?

This research question is meant to explore the possibilities and in some cases increase the efficiency of Grid Manufacturing. Since flexibility by itself is the goal this chapter will not focus on finding optimal efficiency, but more on how to utilise certain aspects of the system in an efficient manner. Note that Grid Manufacturing is dynamic, with changing capabilities of the equiplets due to the reconfigurability and is meant for high-mix, low-volume production. Hence, scheduling is of less importance than with classical manufacturing where the supply and demand is known a priori.

The research questions can be categorised in two categories:

- Proactive aspects - Can agents use top-down (hierarchical) management to optimise production in a pro-active manner?
- Reactive aspects - In reality, some practical factors, like disturbances, occur in the system. These can have a large impact on the efficiency. How can the features of grid manufacturing be used to react to these disturbances and minimise their negative effects?

Seen from these categories we ask the following four sub-questions:

RQ5a How can we validate the system in different cases?

RQ5a1 In what cases is grid manufacturing expected to be implemented?

RQ5b What are the unique factors that influence the production efficiency of a grid?

RQ5b1 What impact does reconfigurability of the production platforms have on production efficiency?

RQ5b2 How do disturbances impact the manufacturing systems?

RQ5c What strategies or optimisations can counter the expected negative factors like disturbances in the manufacturing process?

RQ5d What management strategies can be used to control a grid during various cases?

8.2 Research Objectives

The objectives describe how the answers to the research questions shall be achieved. To investigate the questions simulations will be created so that different aspects of the research can be investigated under controlled conditions. The research will focus on two objectives:

RO1 Investigate the proactive opportunities that Grid Manufacturing can provide.

RO2 Test the efficiency of a grid, including the reconfigurable aspects, and problems that might occur.

The discussion of both objectives will be split into two sections that each discuss their own objective.

8.3 Proactive Top-Down Usage of a Grid

Reconfigurable Manufacturing Systems (RMS) are of interest to industry. The possibility to quickly change a system provides the ability to have a shorter time to market, and scale systems quickly according to current market needs. This flexibility can be approached from different levels, e.g. from modularisation of changeable hardware to the agility of the company. An important aspect in this field is how flexible hardware can be controlled. Traditionally,

manufacturing systems are controlled from a centralised system in a hierarchical manner. A hierarchical or centralised controlled system gives the best efficiency when all systems are predictable. When many disturbances occur, a distributed system with heterarchical aspects (where individual manufacturing systems act autonomously) could possibly achieve better results. Hence, for the production of high-mix, low-volume products, a distributed system, where autonomous systems are not dependent on each other, could be of interest.

Since equiplets are autonomous and can directly interact with products, it is possible to use heterarchical control, where every equiplet is equal to another. The products make use of each service the equiplets provide, depending on the capabilities of the equiplet's configuration. This provides the possibility to manufacture a range of products on one grid of products, offering generic services. Automated manufacturing systems control architectures can be divided into four basic types: centralised, hierarchical, modified hierarchical, and heterarchical (Dilts et al., 1991). This section investigates if manufacturing could be more efficient using some aspects of hierarchical or heterarchical control systems. Since both have their advantages, the possibility to change between these approaches will be discussed, based on the current demand. Different cases will be researched using a simulator where a number of products are manufactured on a grid of equiplets.

In the literature (Leitão, 2009) and (Trentesaux, 2009) present a survey of the intelligent and distributed manufacturing control systems using the emerging paradigms. Both note there is a lack of proven methodology or tools used in industrial practice. Overall trends in various manufacturing sectors are the move from hierarchical management structures to more levelled structures reducing middle management, i.e. moving towards more modularity, autonomy, and self-sufficiency, at the lowest possible levels (Mehrabi et al., 2000).

A good overview of Multi-agent scheduling is given by (Weerdt and Clement, 2009). (Brun and Portioli, 1999) argue that distributed systems have an edge over centralised systems and propose a multi-agent system for simulation of shop-floor scheduling. (Moergestel et al., 2012) proposes a multi-agent system for manufacturing that produces a successful robust schedule against disturbances when it is producing at a high load.

Even though research in this area becomes more available, these contributions are often informal and fragmented. (Maturana et al., 2005) focuses on modelling different agents in a manufacturing domain on JADE, where machines and AGVs are modelled as agents. The application leads to an agent-based simulation with a reactive agent architecture. (Barbosa and Leitao, 2011) state that simulations are crucial in analysing behaviour during the design phase, and present a simulation designed to study agent-based control systems for deployment into real operation.

8.3.1 Simulation

To be able to research different approaches a number of cases will be investigated by developing a simulator that can emulate a set of equiplets in a grid. The simulation is used to determine the efficiency of a grid from a logistical perspective.

The simulation consists of a number of systems and variables that are calculated, which include a mechanism to calculate the time that it takes for a product to travel between equiplets, a product generator, and a planning algorithm. The planning algorithm determines which equiplet can perform the step, based on its capabilities and schedule. This will be discussed later in more detail.

Product Steps

The simulation makes use of equiplets and products. In Grid Manufacturing, the products have a description of how they can be manufactured. This is done by the 'Product Steps', which have been described in Chapter 5. However, in the simulator the product steps are not translated to specific instructions, but are simplified for logistical use, i.e. the simulator checks which equiplet can perform the step and uses this information to be able to schedule the product step. This way the simulator can be used to determine the logistic efficiency of the grid.

In general the same definition for Product Steps is used as defined by (Moergestel, 2014), which defines product steps as follows: "A production step is an action or group of coordinated or coherent actions on a product, to bring the product a step further to its final realisation. The states of the product before and after step is stable, meaning that the time it takes to do the next step is irrelevant and that the product can be transported or temporarily stored between the two steps".

In this sense the manufacturing process of a product can be characterised in the simplest form by a tuple of product steps, e.g. $< \sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5 >$ for a product with 5 product steps. However, while van Moergestel defines the product steps as a standardised set of a predefined steps known by both equiplet and product agents (Moergestel, 2014), this is slightly different in this thesis, i.e. here a product step consists of an abstracted service *and* a set of criteria, as discussed in Chapter 5. The change was a result of the creation of reconfiguration, where different hardware modules were capable of performing the same service, but with different limitations, e.g. two different grippers can both pick up an object, but one might be stronger than the other, being able to pick up a heavier object. Hence, an equiplet does not have a set of standardised predefined product step types it is capable to perform.

Instead, its capabilities are based on a set of services and limitations, which are checked against the service and criteria that are stored in the product step of the product agent. As a result, this chapter will normally show either which service the equiplet is capable of, or which service is requested by a specific product step. For example, a product agent with three product steps could be represented by a tuple: $< \sigma_1\{service_X\}, \sigma_2\{service_Y\}, \sigma_3\{service_Z\} >$.

Simulator Mechanics

The simulator defines a number of equiplets with specific capabilities and products that can be added dynamically over time. In general, a *capability* for an equiplet is defined as the ability to perform the requested service and where the criteria of the product step fall within the limitations of the equiplet.

Transport is taken into account by using average travel time over bidirectional paths between equiplets, using distance information, which is calculated with the following formula:

$$travelTime = (abs(\Delta X_a X_b) + abs(\Delta Y_a Y_b)) \times travelTimePerHop \quad (8.1)$$

Where a and b are equiplets, X, Y are coordinates, and travelTimePerHop is the average travel time between neighbour equiplets.

Other features of the simulator are to add disturbances that delay or stop an equiplet. It is also possible to start production of a batch. A batch is an amount of identical products that are ordered at the same time. The simulator also has a product generator, which can spawn random products that have a random amount of different steps that have to be completed to let it be produced. The product generator that spawns products can be set to deliver a certain grid load. The grid load is calculated by measuring how many equiplets are actually being active during a specific time frame. This makes the product generator take into account which capabilities and equiplets are available to estimate how many products it should create, to stress the grid with a certain 'load'. The spawn system uses a seeded randomiser together with a setting for the chance of delay of the product to make a more realistic (changing) load which can at times create a spike of product demand.

Planning

Grid Manufacturing was originally designed to work with product agents and equiplet agents that are fully cooperative. This makes the entire system *heterarchical*, i.e. all systems are equal and have no planning systems that enforce any hierarchical control. Van Moergestel describes a system where product agents negotiate with the equiplets and are able to control their own manufacturing process (Moergestel, 2014). However, while this might seem logical

for the creation of different products in low volumes, it will likely not be efficient when many similar products are used. Hence, it might be of interest to investigate what happens if not only 'heterarchical' control is used, but also 'hierarchical' control that enforces specific use of all or some equiplets based on a centralised planning algorithm.

To research heterarchical versus hierarchical control, a grid consisting of 9 equiplets is defined. Each of these equiplets has its own set of capabilities. For this section only, the definition of a 'capability' is simplified by ignoring the limitations of the equiplet and the criteria of the product. As such the 'service' which the product agent requests should only be matched with the service that the equiplet can provide. Hence, the equiplet is capable to perform the product step when the requested and provided service match.

Since equiplets are low-cost and single-purpose, the set of capabilities are small, varying from 1 to 3 capabilities per equiplet. In the simulation, simple products are created, based on three actions. To illustrate this we define three different capabilities:

- α for the capability to add adhesive.
- β for rotational operations, i.e. bolting or drilling.
- γ for pick and place operations.

Now consider a product P1, with 4 product steps, defined by a capability that can perform that step: $< \sigma_1\{\gamma\}, \sigma_2\{\alpha\}, \sigma_3\{\gamma\}, \sigma_4\{\beta\} >$

This shows that product step σ_1 can be performed by an equiplet that can perform capability γ, product step σ_2 by an equiplet with capability α, etc.

P1 could for instance be a sensor made out of four product steps. First the electronic parts are placed in a casing, and secondly, adhesive is added to fix the parts. Then the other half of the casing is placed on top of the adhesive, and finally a screw is driven into the thread to fixate the sensor casing. To plan this sequence, a product agent is created that represents the product. The product agent will match all possible equiplets that can perform the necessary production steps that need to be performed, i.e. it compiles a collection, matching the capability of the equiplet with the required product steps. The resulting collection can be shown, where E_x is an equiplet with a unique number; the collection in this case is: $\{E_1\{\sigma_2\}, E_2\{\sigma_4\}, E_3\{\sigma_1, \sigma_3\}, E_4\{\sigma_2\}, E_5\{\sigma_1, \sigma_3\}, E_6\{\sigma_4\}, E_7\{\sigma_2\}, E_8\{\sigma_4\}, E_9\{\sigma_1, \sigma_3\}\}$. Which shows that for this product, Equiplet 1 (E_1) is capable of perform product step 2 (σ_2), Equiplet 2 (E_2) is capable to perform product step 4 (σ_4), Equiplet 3 (E_3) can manufacture product step 1 (σ_1) and product step 3 (σ_3), etc.

In order to optimise production within the grid, batch scheduling is introduced. Batch scheduling allows for reserving a section, i.e. specific number of

equiplets, of the grid to be used only for specific batches. This can be seen as a classical assembly line controlled by a hierarchical entity (likely an agent) that reserves these equiplets for specific use. When a reservation is made, only products spawned in the annotated batch are allowed to schedule with the reserved equiplets. When applying batch scheduling on the simulation, reserving E_1, E_2, and E_3 for batches, the collections become:

Batch production

$$\{E_1\{\sigma_2\}, E_2\{\sigma_4\}, E_3\{\sigma_1, \sigma_3\}\}$$

Other products

$$\{E_4\{\sigma_2\}, E_5\{\sigma_1, \sigma_3\}, E_6\{\sigma_4\}, E_7\{\sigma_2\}, E_8\{\sigma_4\}, E_9\{\sigma_1, \sigma_3\}\}$$

Once these collections have been compiled, the product agent starts negotiating with the equiplets to be scheduled. This system was implemented partly based on the work of Moergestel et al. (2012).

In the implementation of the simulation, the product agent inquires each of the equiplets to evaluate whether or not the step can be performed with the given parameters at the given equiplet. Once all equiplets have been queried and matched, the actual planning begins. The first step in the planning process is to create a production matrix. This matrix describes which equiplet could best be scheduled for which product step of one specific product. In the production matrix the rows represent the equiplets and the columns represent the product steps. The matrix is filled with a score of where a product is best scheduled. The higher the number, the better the choice. The calculation of the number is determined by the following steps:

1. All equiplets that are capable of performing a step have their value raised with 1.

2. Minimise the transitions between equiplets, in some cases one equiplet is able to perform multiple sequential steps. To prevent excess transitions between equiplets during manufacturing, all equiplets with sequential steps have their value raised by the length of the sequence -1. A sequence is defined as the number of steps that the product could perform at the same equiplet.

3. Load balancing. Lower the score for capable equiplets that have a busier schedule. The load of an equiplet is calculated by a % of how fully they are scheduled in the time window that the product step can be performed. The load during this time window is taken into account for the capable equiplets, e.g. when fully scheduled and the equiplet is able to perform 1 product step, i.e when the equiplet has a load of 100% in this timeframe, this would lead to a -1.0 in the matrix, or a 10% load would lead to add a -0.1 score for that equiplet. This will be explained in more detail.

Load Calculation for an Equiplet

To illustrate the planning mechanism, consider a product that starts to be produced at a certain time in the grid, which is called the *release time*, and with a certain *deadline* before it needs to be produced. Now if a product wishes a product step to be scheduled, e.g. product step 4 (σ_4), the product agent queries several equiplets and gives the equiplet agent its deadline and possible arrival time. The arrival time is its release time + Δ, where Δ is the planned extra time that it will take the product to have the first three product steps ($\sigma_1, \sigma_2, \sigma_3$) executed plus the travel time it will take to the new equiplet. This time period from the time it can arrive, i.e. releasetime + Δ) until the product deadline, is called a 'time window' or 'time frame' in which the product step can be performed.

To choose at which equiplet the product step of the example is produced is partly based on how 'busy' an equiplet is during this time window. This is shown by the 'load' of an equiplet. The load is the percentage of time that product steps are scheduled in this time window, e.g. if an equiplet is planned for 4 timeslots out of 10, during a specific time window, its planned load is 40%. A product can query several equiplets with the correct capabilities to calculate the planned load for these equiplets at the selected time window.

Figure 8.1 shows an example of how the schedule for two equiplets could look like. Showing when the equiplet has certain products planned.

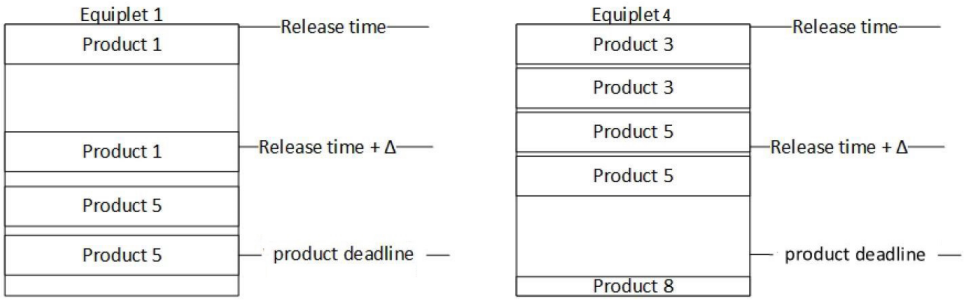


Figure 8.1: Two examples of equiplet schedules.

In this example equiplet 1 has nearly no free time in the requested time window, which starts at arrival time (release time + Δ) and ends at the deadline of the product. The equiplet will therefore communicate the expected load, based on its planning, during the time window that the product requested. In this case, equiplet 1 will respond that the expected load is 90% during the

Table 8.1: An example production matrix with load balancing, for a product with 4 products steps, in a grid of 9 equiplets. Best candidates are shown in **bold**.

Equiplet{capability}	σ_1	σ_2	σ_3	σ_4
$E_1\{\alpha\}$	0.0	0.4	0.0	0.1
$E_2\{\beta\}$	0.0	0.0	0.0	0.4
$E_3\{\gamma\}$	0.4	0.0	0.4	0.0
$E_4\{\alpha\}$	0.0	**0.8**	0.0	0.0
$E_5\{\gamma\}$	**1.0**	0.0	0.8	0.0
$E_6\{\beta\}$	0.0	0.0	0.0	**0.6**
$E_7\{\alpha\}$	0.0	0.6	0.0	0.0
$E_8\{\beta\}$	0.0	0.0	0.0	**0.6**
$E_9\{\gamma\}$	**1.0**	0.0	**0.9**	0.0

requested time window. However, while equiplet 2 is busier in general, it actually has a lower load in the requested time window. Hence, equiplet 2 will tell the product it has a 30% load during the requested window. As such the product will favour E_2, even though that equiplet has a higher overall load.

During normal operation, these steps are performed for all steps and equiplets. The load windows are added to the production matrix by multiplying the numbers based on capability and multiplying these with $(1 - load)$. This could for example result in a production matrix as shown in Table 8.1.

When using load balancing, Equiplet 1 (E_1), has a score of 0.4 for step 2, as can be seen in 8.1, this says it has a load of 60% during the time window that the step could be planned. Hence, due to load balancing Equiplet 1 is suddenly not the best option for step 2 any more, since instead of both equiplets 1 and 4 having a score of 1.0 (without load balancing), now Equiplet 4 has only a 20% load during this time (a higher score of 0.8, compared to 0.4 of equiplet 1). While other options are possible, this system does provide ways to load balance between the equiplets, by giving a better score for equiplets that have a lower load during the timewindow in which a product step could possibly be performed.

Another optimisation could be to reduce the amount of time a product travels within the grid. This can be achieved by calculating the different paths through the grid over all steps, and applying these values to the production matrix. Because the travel method was not designed yet at this time and as such could not give a realistic schedule, optimising travel time was not taken into account for the current schedule results.

8.3.2 Top Down Management

It is possible to manage the grid in different ways, by adapting its standard heterarchical structure, this is performed in this simulation using multiple cases which utilise different management strategies. The simulation presents two different cases, which are used as a proof of concept for these strategies.

- Case 1 will show a heterarchical strategy vs a hierarchical strategy, by using an entity that reserves equiplets for batch production
- Case 2 will show what happens when switching from hierarchical control to heterarchical control, including the use of disturbances, which might happen in a realistic scenario when the planning might not always be accurate.

The simulation is performed using products that use three different capabilities, and is performed over time, which is shown in timeslots. Every product step takes a different time to perform. It takes 20 timeslots (relative period) to use capability α, 10 timeslots to use capability β, and 5 timeslots to perform capability γ.

Case 1A - Heterarchical Control

Figure 8.2 on the left shows a grid of equiplets that will be used in the first case. In case 1 we try to investigate the difference between Grid A, as shown in Figure 8.2 on the left, and Grid B, as shown in figure 8.2 on the right. The only difference between these two grids is that with type B, three equiplets, shown in orange, are specifically reserved for product batches. In this case these reserved equiplets are not optimised, i.e. specifically configured, for the batch product that is used in the simulation.

In case A, all equiplets are 'equal', i.e. set for heterarchical control, and all equiplets can be used by any product. To simulate these, 351 timeslots (which represent 2 hours) are simulated, spawning 35000 random products, of which 3600 products in a batch. The products use all three capabilities that are available in the grid, and have a deadline of 86 seconds to be completed. This deadline and the amount of products are chosen such as that the grid performs at an estimated 80% average load, using the random product spawn system. Figure 8.3 shows the load of all 9 equiplets.

Figure 8.3 shows, as expected, a high load on all equiplets. Basically the equiplets with capability α have the highest load, which could be expected since this action takes the longest to perform. This is made even clearer in Table 8.2, which shows the average load over the shown time period for each equiplet and the three equiplets with the same capability. This will be analysed further after the next case.

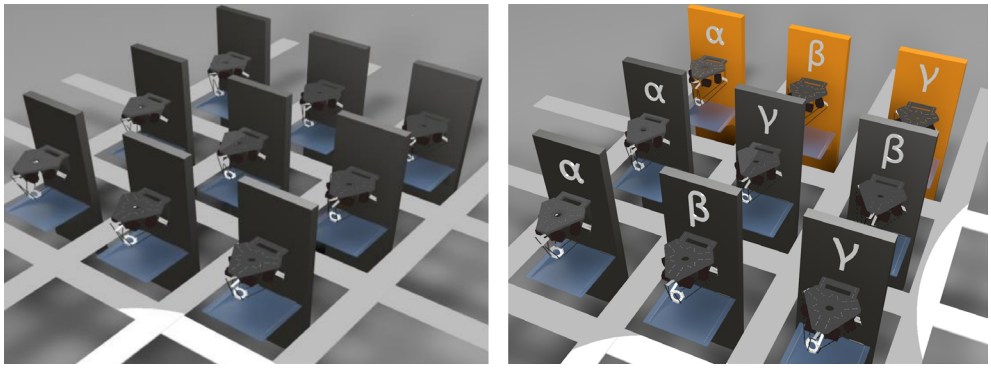


Figure 8.2: On the left a type A 3 X 3 grid with heterarchical control and on the right Grid B, also showing capabilities and the reserved equiplets for batch processing in *orange*. While only B shows capabilities, they are the same for both.

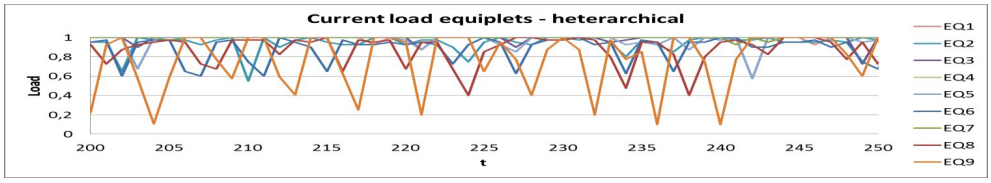


Figure 8.3: The load (1 represents 100%) of 9 equiplets in case 1A.

Case 1B - Hierarchical Control Through Reservation of Equiplets for Batches

In case 1B we look at the same result while spawning products around an estimated 80% load average, but then using three equiplets that are reserved specifically for the batches

Figure 8.4 and table 8.3 show the load in this case. Equiplet 1, 2 and 3 are now reserved for batches. Equiplet 1 has a 100% load, as was also the case in case 1A. This can be explained, since the α capability takes the longest to perform. However, all other equiplets (both reserved and non-reserved) have a lower average load. Figure 8.4 also shows that the reserved equiplets have a stable load, since they are continuously manufacturing identical batch products.

Table 8.2: Average load during shown time period, per equiplet and capability.

E_1	E_2	E_3	E_4	E_5	E_6	E_7	E_8	E_9	α	β	γ
1.00	0.94	0.99	1.00	0.97	0.89	1.00	0.88	0.77	1.00	0.90	0.91

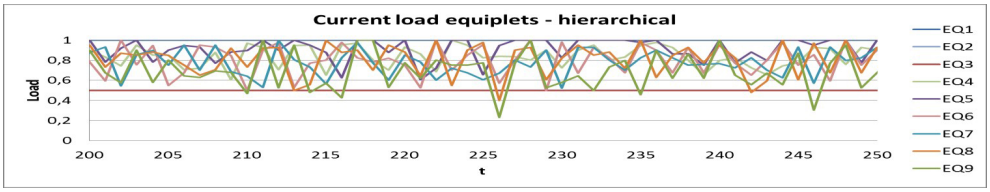


Figure 8.4: The load of 9 equiplets in case 1B with reserved equiplets for batches.

Table 8.3: Average load during shown time period, per equiplet and capability.

E_1	E_2	E_3	E_4	E_5	E_6	E_7	E_8	E_9	α	β	γ
1.0	0.50	0.50	0.84	0.91	0.80	0.77	0.81	0.69	0.87	0.70	0.70

Analysis case 1

The results show that under the current conditions, reserving equiplets provides a stable, but also lower, load for the non-reserved equiplets compared to not using reserved equiplets. The cause of this is the lower choice of free equiplets to the random products, which leads to fewer possibilities where the products can be manufactured in time and an expected lower throughput (finished products over time). However, as shown in Figure 8.5 there are more differences. Figure 8.5A, at the top, shows that the number of active products in the grid is substantially higher when no equiplets are reserved, while throughput, as shown in Figure 8.5B in the middle, is just slightly lower. This can be explained by a shorter travel distance and therefore a faster completed product per equiplet. Figure 8.5C on the bottom also shows that some equiplets were overloaded when using heterarchical control, which led to more *failed products*, i.e. products that were unable to reach their deadline.

It can be concluded that heterarchical control has a potential higher throughput in some cases, at the cost of production time per equiplet and potential overload of the equiplets. Also, in a practical case using real equiplets, the higher amount of active products in a grid with heterarchical control might pose challenges for logistics, with possible delays due to the longer travel time.

Case 1 shows that both hierarchical control, where an entity reserves some equiplets for batches, and heterarchical control, when all equiplets can be equally used by any product, can have benefits. Hierarchical gives less chance of failed products, a more predictable load on the system if there are many batches, and has less active products in the grid. Hence, heterarchical control has a potentially higher throughput when the reserved equiplets are not optimised for the batch product. This gives the grid the ability to manufacture more products in the presented case at the cost of some products not meeting

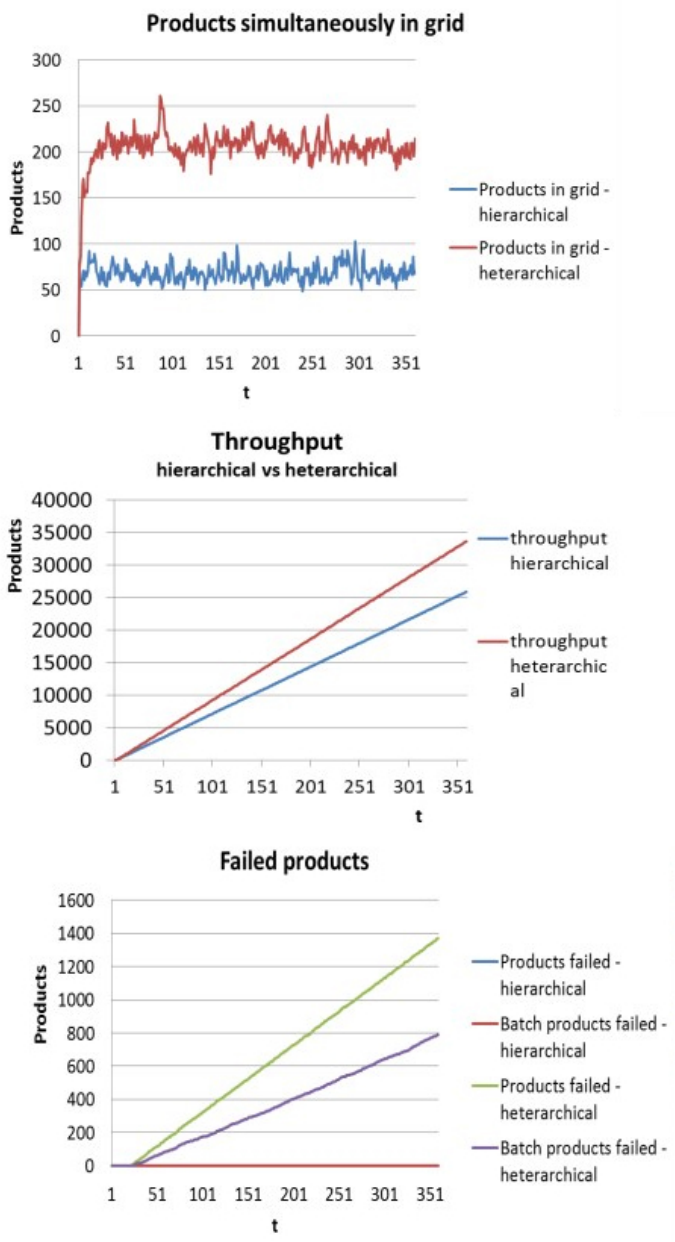


Figure 8.5: A-top: Active products in grid. B-middle: throughput (completed products). C-bottom: failed products that did not meet their deadline.

their deadline and predictability.

What was not shown is a test case where reserved equiplets would be

reconfigured and chosen based exactly on the needs of the batch product, since the results would be known beforehand. When there is enough demand for the batch product, the reserved equiplets would run near 100% capacity, while the other 6 equiplets would run like a standard grid of 6 heterarchical equiplets.

The opportunity for this lies in the ability to quickly reconfigure equiplets and reserve them when large batches are expected. This basically creates the ability to use the high throughput for different (random) products and change to the efficiency of standard line manufacturing when large batches of products are expected to be made.

Case 2 - Error Behaviour - Switching From Hierarchical Reserved to Heterarchical

Case 1 showed that both hierarchical control and heterarchical control with reserved equiplets have different advantages. However, it is evident that disturbances will have a larger effect on hierarchical control. As with any line, if the reserved equiplets encounter any kind of disturbance the batches will stop producing, which leads to the failure of all batch products until the problem that initiated the disturbance can be resolved. However, since equiplets can be reserved by a hierarchical entity, but are designed to be autonomous, it is of interest to investigate if switching from hierarchical to heterarchical provides the ability to mitigate these disturbances. This could also open future research for dynamic reservations when large batches are expected.

To investigate this case we take the same simulation settings as in the previous case. This means an average of 80% load is maintained on the grid by using random products, whilst spawning 1 batch product every 2 seconds. The same grid was used as in case 1b, as was shown in Figure 8.2 on the right. After timeslot 180, equiplet 3 encounters a disturbance and goes into an error state which renders it unavailable for further production. As a result of the disturbance, all reserved equiplets that are able (without error) will change to a heterarchical state. The batches will also be able to make use of all available equiplets.

At figure 8.6A the disturbance that stops equiplet 3, with capability γ, can be clearly seen. This would normally stop all batch production. However, in this case equiplet 1 and 2 are immediately switched to heterarchical mode and batches are allowed to be rescheduled at any available equiplet in the grid. While this could potentially lower efficiency (depending if the reserved equiplets are efficiently chosen for the batch products), the problems of the disturbance are mitigated since equiplet 5 and 9 immediately compensate for the disturbance. This can clearly be seen in Figure 8.6C, which shows equiplets 5 and 9, which have a similar capability as equiplet 3, immediately increase

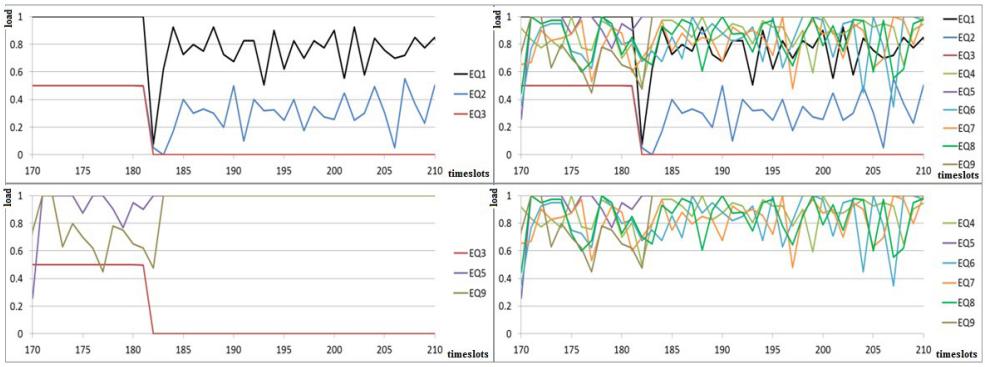


Figure 8.6: A-top left: load of the reserved equiplets, B-top right: load of all equiplets. C-bottom left: all equiplets with capability γ, D-bottom right: original heterarchical equiplets.

their load to a 100%. As shown in Figure 8.6D the impact on the equiplets that were originally heterarchical is only slightly noticeable, since the overall load is similar in comparison to the load before the disturbance.

Case 2 Analysis

Case 2 shows the use of being able to switch back to heterarchical manufacturing. Figure 8.7 shows how many products would be unable to reach their deadline if the batches were not allowed to change to the heterarchical system and if they would be allowed to.

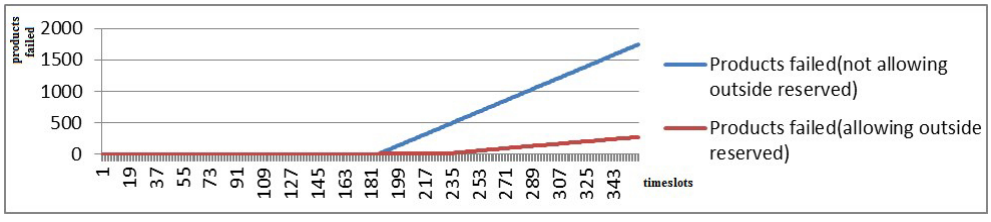


Figure 8.7: Shows the amount of failed products.

As expected, many more products fail after the disturbance occurs. Switching to heterarchical mode clearly provides a lower amount of products that are unable to meet their deadlines. However, some products still fail after the disturbance due to the increased travel time and higher load of the other equiplets that have to take over from the disabled equiplet.

8.3.3 Discussion

Case 1 might be considered slightly arbitrary, since the results are highly dependent on a large number of variables including: size of grid, composition of capabilities, required steps for the product, the deadline and their randomised starting point. However, this was the intended point. In grid manufacturing, products can be dynamically added to the grid and the simulation shows that grid manufacturing can deliver this large range of possibilities. Since equiplets are also reconfigurable in nature, it would be possible to quickly reconfigure the hardware to accommodate large batches of identical products and form an efficient line using the hierarchical entity that reserves equiplets for these products. This would even be substantially more efficient than shown in case 1B, since here the reserved equiplets were not specifically updated for the used batch product.

While the reserving of equiplets does make the grid more susceptible to error, case 2 proves that the possibility to switch back to full heterarchical use of all equiplets provides the ability to deal with disturbances. Considering grid manufacturing was originally intended as a fully heterarchical way of manufacturing, adding the hierarchical entity for reserving batches provides many interesting possibilities.

8.3.4 Conclusion for Proactive - Top-down Approach

To answer the original research questions: this study shows that there are cases that could benefit from both hierarchical and heterarchical strategies. Based on the needs (size of batches, acceptable failure to meet the deadline and the grid capabilities) choices can be made which strategy to choose. However, disturbances do have a high impact on batch production, which can be mitigated through switching back to heterarchical control. Hence, the conclusion is that it is of high interest to consider both the heterarchical and hierarchical approaches to optimally utilise the possibilities of grid manufacturing.

This section impacts the way autonomous manufacturing systems are used in a grid. While the original goal of grid manufacturing was to create the maximum amount of flexibility, this section shows that it is important to be able to limit this flexibility to create a higher efficiency for any given situation. While this might seem intuitively right, no systems in manufacturing are known to utilise this possibility. Hence, it can be concluded that the development of a hierarchical entity that can reserve equiplets for batch production is a valid and interesting research subject for the future.

Considering the results of this section, and the generic possibilities that grid manufacturing provides, it would be of interest to continue the research in future work. Several steps that should be taken are: 1. Analyse a large amount

of cases to find a heuristic that could determine when reserving equiplets would be beneficial. 2. Use the heuristic to develop an automatic system that could select equiplets to be reserved when this would be beneficial. 3. Integrate this system into the real (physical) grid using agent technology to test the practical implications of the proposed system.

8.4 Reactive Aspects

This section focuses not on the proactive management in a hierarchical aspect as the last chapter, but on how the grid manufacturing system could best be utilised on a reactive basis, e.g. when a disturbance occurs, like when an equiplet breaks down. It also investigates the effect and possibilities that reconfiguration of the equiplets will provide.

8.4.1 Problem Description

Moergestel et al. (2012) researched scheduling mechanisms in an agent-based manufacturing system. In this section we add the ability for manufacturing systems to be reconfigurable, i.e. changed to provide different services. This will change the capacity of the capabilities offered by the grid without interference to other manufacturing systems. Both the dynamic start of new (types) of products, changing (dynamic) job demand and the (reconfigurable) machines themselves create possibilities to optimise the manufacturing process.

Within a system where both the product demands and manufacturing capacity are dynamic, it becomes difficult to use classic scheduling. This becomes an even larger problem if disturbances are taken into account, which are likely in a realistic scenario. Since products have to be transported with a dynamic path and their location is also determined by computer vision, it becomes increasingly hard to plan a schedule correctly. Hence, a new control strategy is required to deal with these circumstances. This is also supported by similar work. (Trentesaux, 2009) notes that it is important to research industrial practices for intelligent and distributed manufacturing control systems. (Barbosa and Leitao, 2011) and (Leitão, 2009) mentions that simulations are crucial in analysing behaviour during the design phase and presents a simulation designed of studying agent-based control systems for deployment into real operation. (Duffie and Prabhu, 1994) identify that an agent-based architecture can be a solution to adapt to production disturbances in manufacturing systems in real time. This brings us to the research questions for this section.

1. RQ5b1 What impact do reconfigurable manufacturing systems have on production efficiency for high-mix, low-volume production?

2. RQ5b2 How do disturbances impact dynamic manufacturing systems?

3. RQ5c What strategies can counter the effect of disturbances?

8.4.2 Platform Description

The REXOS platform as described in Chapter 6 is used to answer these research questions. To be able to understand the approach, some aspects of REXOS will be explained here. This includes the Product Step Matching algorithm that considers if the (abstract) description of a physical action in the manufacturing process, the product step, can be performed by a specific equiplet. The matching of the product steps that the product needs to be manufactured and the equiplet that is capable to perform them is done as follows:

Algorithm 1 Product Step Matching.

for all $\sigma \in P_i$ **do**
  $(s, c) \leftarrow \sigma$
  $E \leftarrow$ SEARCHEQUIPLETS(s)
  $E_{capable} := \{\}$
  for all $e \in E$ **do**
    if $C_e(\sigma)$ **then**
      $E_{capable} := E_{capable} \cup \{e\}$
    end if
  end for
end for
$travelTimes :=$ RETREIVETRAVELTIMES$(E_{capable})$
$productionPath :=$ SCHEDULE$(E_{capable}, travelTimes)$

Where P_i is the product that will try to find the equiplets that are capable to execute its product steps (σ). The s is the service and c the criteria of the product step. The subroutine *searchEquiplets* returns a set of equiplets that provide the services given by the Directory Facilitator. After this the set$E_capable$ is cleared. The C_e provides a set of capabilities for all equiplets; this is checked for every product step to see if it can perform the service s with criteria c. In reality this is implemented as a subroutine that checks the service and criteria of the product step against the provided service and limitations of the equiplet to determine if it is 'capable' to perform the step. If the equiplet is capable to perform the step it is added to the $E_{capable}$ set. After that the traveltime and productionPath is determined by the retreiveTravelTime and schedule subroutines.

8.4.3 Hypothesis

In the current approach each equiplet is set to act as equal (heterarchical) (Telgen et al., 2014), i.e. all equiplets and products act autonomously without a higher form of management and/or optimisation. An event-based simulator has been developed that runs alongside REXOS and simulates external events to test several cases. The hypothesis is that when manufacturing a diversity of products the grid will be able to adapt to the production need and as such can be used more efficiently. However, there are many influences on the efficiency of a reconfigurable system.

In the current section the following cases are investigated:

- Reconfigurable vs. non-reconfigurable systems
- Influence of variable (stochastic) processing times for product completion
- Effect of Rescheduling and Queue Jumping
- Influence of manufacturing system breakdowns

8.4.4 Mechanics

In Grid Manufacturing a number of processes and mechanics can be used.

Systems are reconfigured by a *reconfiguration procedure*. The reconfiguration procedure changes the equiplet with the lowest load towards an equiplet type that has the highest load. The procedure is only performed when there are two equiplets of the same type with the lowest cumulative load under a certain threshold.

Variable (stochastic) processing time stands for the time that is required for the product to be completed. Breakdowns are occurrences that delay or stop the manufacturing process at one equiplet, e.g. when a mechanic has to clear a jam of the filament at a 3D printer (like a paper jam in a normal printer, both happen quite often in reality). It would be ideal to predict the exact processing and travel times such that there will never be any disturbance in the schedules. In real situations, variances in processing times always occur. These can either happen because of the dynamic nature of a manufacturing grid, that is to say: production time can take longer since the exact processing time is unknown, i.e. it cannot be exactly predicted when a product step for this specific product performed at a specific equiplet using a specific configuration is completed. Or, problems can occur during transport or breakdowns of the equipment.

Rescheduling is a procedure in which a product will reschedule its remaining product steps, i.e. release the time slots and re-initiates its scheduling behaviour. It will reschedule whenever the product agent notices that the product step will not arrive at the next equiplet in time.

Queue Jumping is a procedure where products that are available, i.e. have arrived, at the equiplet can be started before they are scheduled. This can happen in the case that the next scheduled product has not yet arrived, but a product which is expected at a later time is already available before it is scheduled. Queue Jumping might be an important improvement since products that are not finished in time will delay all other products in the queue. It can result in a cascade of equiplets that are waiting for their scheduled, but delayed, products to arrive. Queue jumping solves this problem by giving products that have already arrived priority over products that are scheduled to start next, but have not arrived. The product that is too late is placed behind the current product that starts. Which might result in products that are not completed within their expected completion time. These are called *overdue* products.

8.4.5 Simulation Model

A discrete-event simulation model (Kelton and Law, 2000) is developed for the purpose of investigating the cases. After each occurring event the state of the system changes to a new situation.

Assumptions

The following assumptions are made about the grid: equiplets only know the average time it takes to perform a job and not the exact time *a priori*. Variable processing times are decided using an average processing time for a specific type of product step with an offset based on an exponential distribution. Equiplets can either break down or require maintenance; there is no distinction within the simulation between those two. All equiplets have an identical chance of breaking down and an equal average time required to be repaired. No distinction is made between the causes of down time. Storage and transport capacity are not taken into account. Travelling between equiplets always succeeds and takes a certain standardised time per distance to be covered.

The order in a production path is important, as to manufacture a product its product steps are dependent on each other. We assume that each product step requires the completion of the step before, i.e. there is no parallel production for a single product.

Events

There are 7 different events generated for the simulation: product creation, product arrived by equiplet, product started, equiplet finished with a job, equiplet breakdown, equiplet is repaired, reconfigured, and a done event. The transitions and events between these states are described in Figure 8.8.

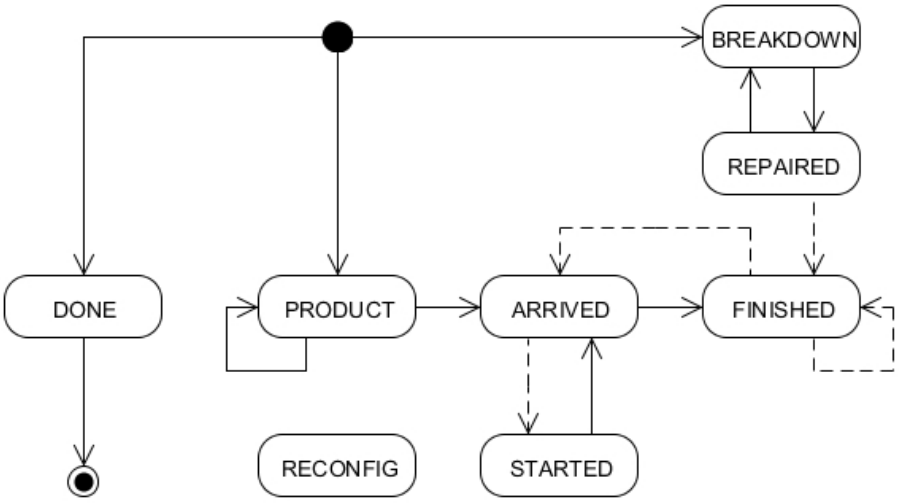


Figure 8.8: All simulation events.

- The first event that occurs when starting the simulation is a *product creation event*. The event triggers the creation of a product agent with a certain set of product steps and a deadline. When the agent manages to schedule his product steps, a product arrived event is added to the event stack.

- A *product arrived event* will trigger the product agent to let it know it has arrived at the scheduled equiplet.

- A *started event* is triggered at the latest time that the product should have been started. If this is not the case, the product knows that it has to reschedule the remaining product steps.

- An equiplet is finished with a product step. It informs the relevant product agent with the *finished event*. If the product travels to the next equiplet, the simulation will schedule a product arrived event.

- The simulation gets a *breakdown event* when an equiplet breaks down or is in need of maintenance.

- The simulation gets a *repaired event* when the equiplet can continue its activities. After an equiplet has been repaired a new breakdown event is scheduled.

- A *reconfig event* is scheduled when the equiplet can be shut down and reconfigured into a new configuration. The event will trigger the change of capabilities and therefore to register its new services with the Directory Facilitator.

- The *done event* will end the simulation. The required procedures such as saving the statistics and taking down the living agents are executed. The planning for this event is performed at the beginning of the simulation to ensure a specific runtime.

Simulation - Equiplet Interfaces

Interfaces were made to simulate external or internal events. For the equiplet the state transitions caused by these are shown in Figure 8.9.

Each of the simulation events needs to be completely handled before handling the next event. This would be different in a real-time system. The equiplet can be in the idle, busy or error state. Besides the hardware states, the equiplet also has three additional simulation states: ready to execute a job as a product arrived while the equiplet is broken down; the equiplet should have been finished with the job but has broken down in the meantime; and the equiplet is repaired after having broken down but does not know when the job would have been finished. As it is not possible to look into the future, equiplets need to make a distinction between the state 'busy', when they are active and performing a job, or when it continues with a job after having been broken down. This to change the according time that a product is completed, since a breakdown will delay the completion time. The same goes for the distinction in the error state, when the equiplet would have been finished or when a product arrived while being broken.

Case Parameters

The current cases are performed with a 24 equiplet setup, with 4 different capabilities. The setup has an even distribution of capabilities. However, the time it takes to complete a product step is different. In this case the average production time for $\sigma_1 = 15, \sigma_2 = 15, \sigma_3 = 30, \sigma_4 = 120$. As is common in manufacturing simulations the time between product arrivals is calculated using an exponential (Poisson) distribution with a random mean variable. In the simulation target utilisations are used to determine how many products should be created at the grid. A 'product spawn' system creates new products based on the target utilisation. A 100% target utilisation would mean that the equiplet under ideal conditions would have enough products to be working all the time. The time between arrivals, (inter-arrival times) of products follows a Poisson distribution with a mean depending on the target utilisation of the

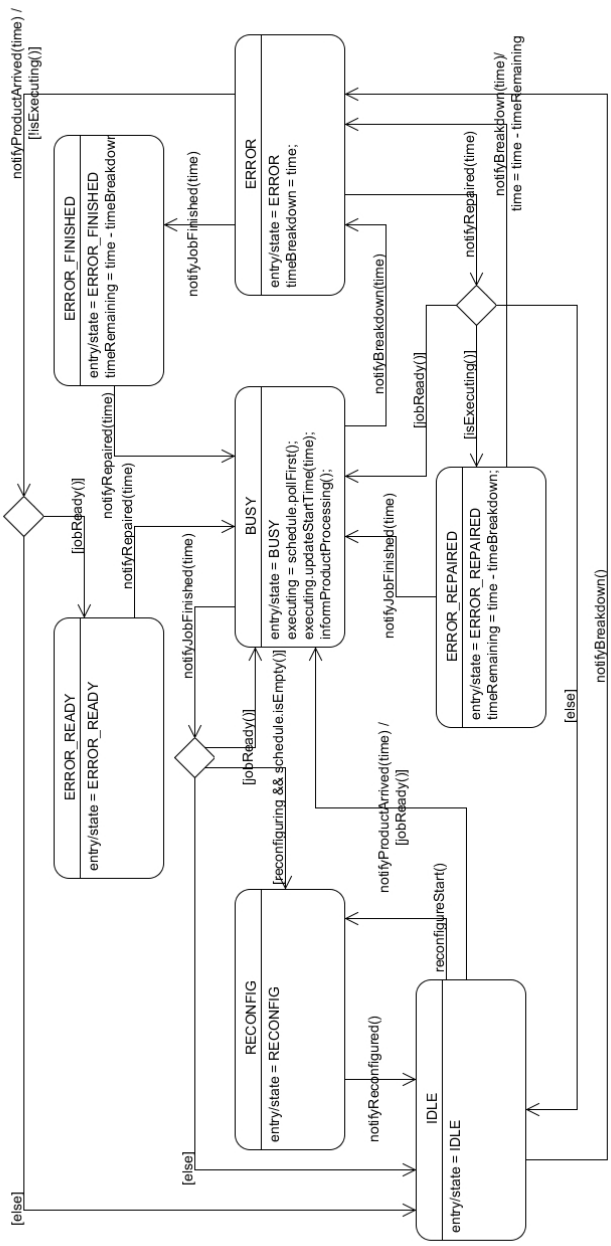


Figure 8.9: State Transition Diagram for equiplets.

grid. Where the utilisation ρ is the expected service time E(S) divided by the expected inter-arrival time. For the current case the optimal utilisation

$\rho = 0.8$ is used, based on (Moergestel et al., 2012).

$$\rho = \frac{E(S)}{E(A)} \tag{8.2}$$

Each product spawned by the simulation has 20 random product steps. The expected service time of a product is the average production time times the number of product steps. For each run there is a start-up period that is not taken into account for the results; this is done to simulate a continuous production environment.

The reconfiguration procedure can be started when two equiplets with the same capability have a cumulative load under 110%. If the cumulative load is over 110% it is not possible to reconfigure, since this will lead to a large shortage of equiplets that can perform the same specific capability. If the reconfiguration is triggered, the equiplet with the lowest load will change its reconfiguration to a configuration that is able to perform the same capability as the equiplet with the highest load in the grid.

8.4.6 Implementation

The simulation has a Graphical User Interface (GUI) which shows a large amount of information. Including the status of every equiplet, products in the grid, statistics, etc. Figure 8.10 shows the GUI for the simulation.

8.4.7 Results

All data is based on the average result of 10 runs. The results are given in two different subsections, first to describe the effects of reconfiguration, and second the effects of disturbances in the grid.

The Effect of Reconfiguration

Shown in Table 8.4 is the average load of the entire grid (average of all equiplets), the amount of products that have been finished in the time period. Also shown is the average time it takes for a product to be finished from its start, this is called the *Production Time*, and the amount of products that have failed to schedule. Failure happens in cases that the product is unable to create a schedule that finishes before the product deadline.

The results show that when producing a variety of products, reconfiguring the manufacturing machines proves to be much more efficient. This was expected, since product steps take a different amount of time to complete, and as such a bottleneck will likely occur. However, when systems can be reconfigured the grid will automatically adapt to the demand by changing its least used equiplet into one with a configuration that has a high demand. This

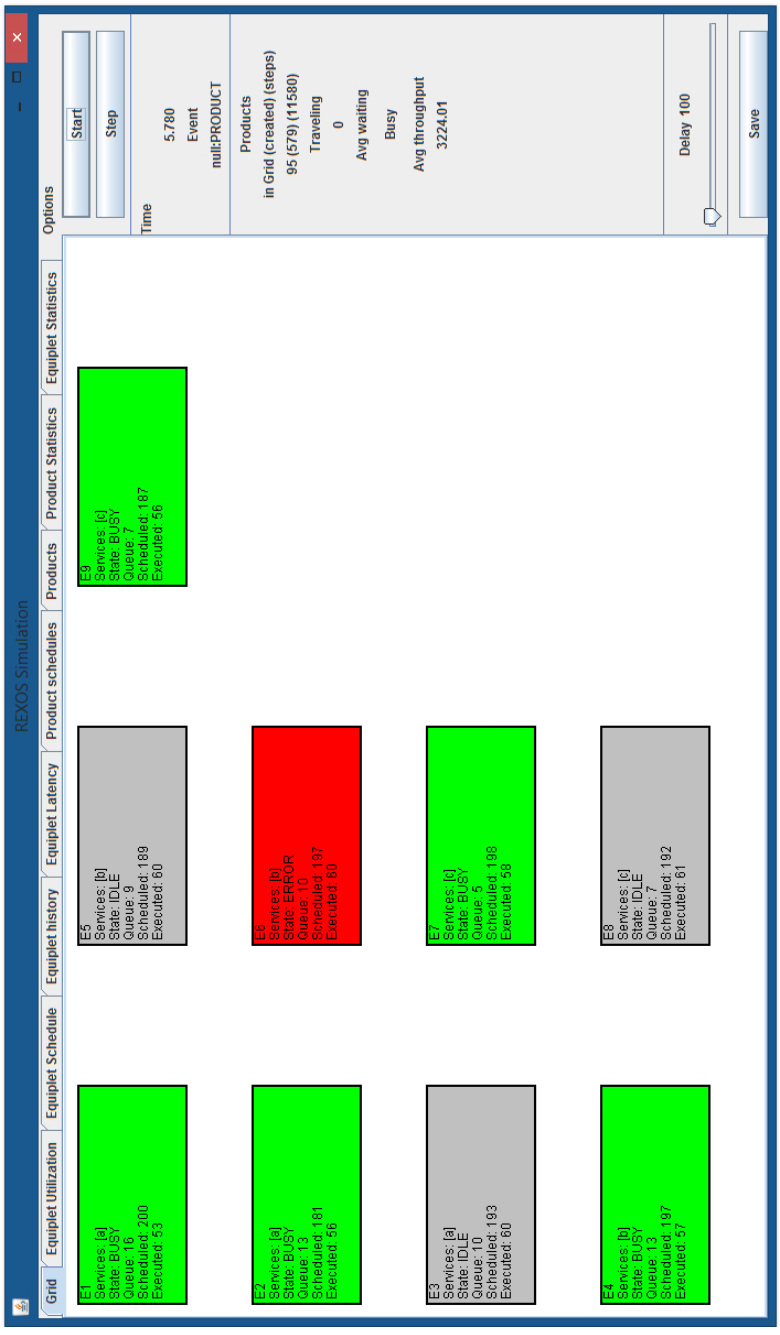


Figure 8.10: The GUI of the Simulation, showing multiple equiplets.

leads to the completion of more products and less products that are unable to be created before their deadline.

Table 8.4: non-reconfigurable vs. reconfigurable systems.

	Load	Products Finished	Average Production Time	Products Failed to Schedule
Base	35%	10637	9546	10631
Reconfig	77%	20636	5879	617

Figure 8.11 shows the result over time, by showing the average load of all equiplets. Clearly the reconfigurable systems adapt with the demand and saturates to a 77% load, which gives a 94% increase in products finished. Drops in load with the reconfigurable systems are due to the changeover time when an equiplet has to be reconfigured to be able to provide another capability to the products.

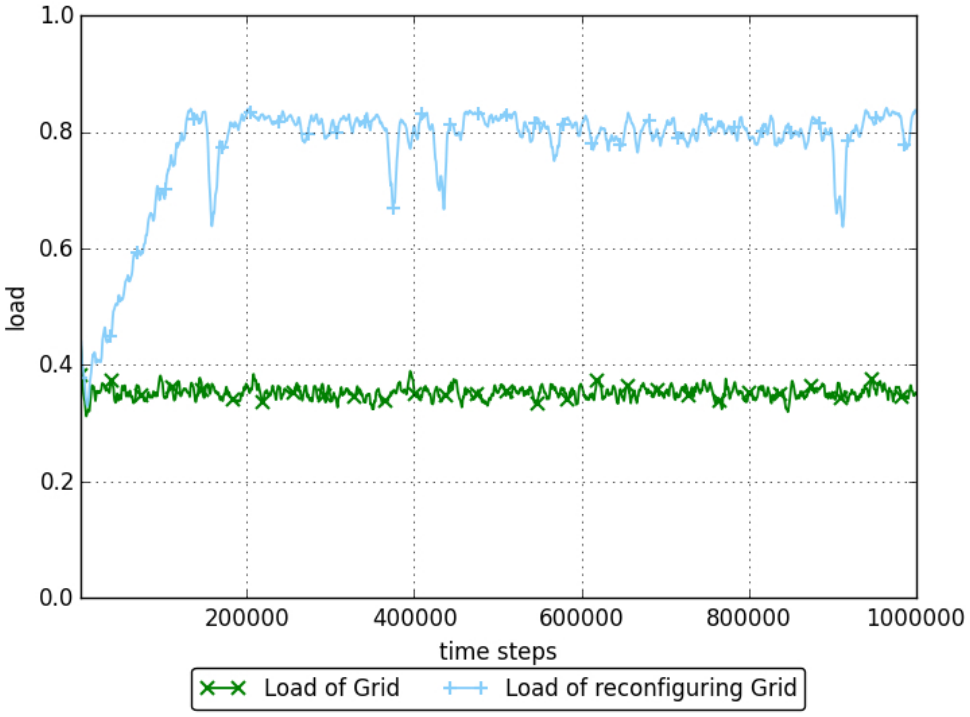


Figure 8.11: Results of one run with none vs. reconfigurable systems over time.

When the grid reaches the theoretical optimum (for the average product) there are still changes in configuration and therefore capabilities, leading to a reduced utilisation of the grid. This is due to the randomness of the re-

quired capabilities of the products. At times it might seem more efficient to change the configuration of an equiplet, but after the reconfiguration occurs the demand is changed back in a way that the original configuration was more effective. Since the demand and configuration keep changing based on the current demand it might keep changing all the time. However, using statistics it might be possible to more accurately predict the efficiency of a reconfiguration action in the future.

It is possible that the product fails to schedule all of its product steps, i.e. it does not enter the grid due to the high utilisation that makes it impossible to schedule all steps within the deadline. In this case using reconfiguration, products that are unable to schedule within their deadline are limited to 2.9%. It can also occur that a product fails to meet the deadline; this is because rescheduling keeps occurring due to a changing demand, which could lead to delays.

The Effect of Disturbances

To show a more realistic case, the effect of disturbances have to be taken account. Hence, the effects of *stochastic/variable completion time*, and equiplet *breakdowns* will be added to the same simulation cases.

Table 8.5 shows the effect of disturbances with three different cases, which are all reconfigurable. This is compared with only stochastics turned on, as well as with stochastics and breakdowns turned on. To counter the effects of these disturbances they are shown with the effect of a base setting (no rescheduling or queue jumping), rescheduling turned on, and queue jumping turned on together with rescheduling. Stochastic production completion and breakdowns also introduce a new effect, i.e. products that can be scheduled within their deadline, but are unable to be completed within their scheduled time. These are considered overdue. Both breakdowns and the stochastic completion time greatly affect the amount of products that can be finished. Together they halve the amount of products that can be completed in the current case. Both queue jumping and rescheduling counter these effects, almost doubling the amount of products that can be manufactured within the same time frame when these disturbances occur.

The impact of stochastics and breakdowns prove to bring the products created under the level of the original base configuration (without reconfiguration). Rescheduling brings the load from 35% to 57% and queue jumping brings this up to 73%.

Table 8.5: reconfiguration results.

	Load			Products Finished		
Queue Jumping	**base**	**resched.**	**qj**	**base**	**resched.**	**qj**
Reconfig	77%	77%	79%	20636	20685	20924
Reconfig + Stochastics	36%	59%	76%	9693	16380	20830
Reconfig + Stoch. + breakdowns	35%	54%	73%	9295	15089	19654
	Production Time			**to Schedule**		
Reconfig	5879	15089	3283	617	578	397
Reconfig + Stochastics	11224	17161	5508	11567	4691	434
Reconfig + Stoch. + breakdowns	11593	16940	8383	11971	6024	1540
	Overdue					
Reconfig	0	0	101			
Reconfig + Stochastics	5879	14063	138			
sReconfig + Stoch. + breakdowns	5759	12608	607			

8.4.8 Discussion

The cases in this Chapter are not meant as validation for overall efficiency, since that would require many more cases. However, they are used as a proof of concept to show how Grid Manufacturing can be utilised and to provide solutions to new unknown factors that come with the flexibility of Grid Manufacturing. Dynamic production for high-mix, low-volume is a difficult topic to research. Most papers simplify this approach by removing unknown factors like disturbances and the variation of the production time that likely occurs when using reconfigurable machines with similar capabilities (Brun and Portioli, 1999; Komma et al., 2011; Barbosa and Leitao, 2011). In the current study these are taken into account to validate a more realistic proof of concept that is made possible by the abilities of grid manufacturing. This is important, because when dynamically producing a range of products the effect of disturbances proves to be substantial. However, it is a challenge to provide realistic parameters. Since both the number and type of products are random, it is difficult to optimise the configuration of the manufacturing systems to an optimum. This also influences the comparison between reconfigurable and non-reconfigurable systems. Clearly reconfigurable systems will have the ability to better adapt to the current demand. While this is viable for a case with random products, results can vary strongly based on the case used, e.g. cases where batches are produced. However, the results show that it is likely that reconfigurable systems will prove more effective when dynamically manufacturing high-mix, low-volume products. So while this simulation on top of the control system for grid manufacturing clearly proves more efficient it is more interesting to see the effects of other disturbances and the strategies that counter these, which was shown with queue jumping and reconfiguration. Especially stochastic production completion has a large impact on the products that can be completed. This is because of a number of factors, including the unknown hardware where the product will be produced, the speed with

which the vision system will find the arrived product, and disturbances that will occur in the manufacturing process itself. As such, it is impossible to exactly follow the schedule, which will have a large impact on other products as well. Rescheduling and queue jumping minimise these effects.

Basically, the goal of the current chapter was not so much to prove the efficiency to the grid, but to show the possibilities of grid manufacturing and how to deal with practical problems that occur when using such a dynamic production system.

8.4.9 Conclusion Reactive

Grid manufacturing opens the possibility to dynamically manufacture a range of high-mix low-volume products. However, this type of manufacturing also introduces new effects that greatly affect the overall performance. Reconfigurable systems can be efficient when manufacturing a large variety of products at random. However, in a real case the impact of disturbances becomes very important. Disturbances show to have a high impact, because of the stochastic and dynamic behaviour of the grid. The current section shows that it is important to use strategies that adapt to these disturbances. The disturbance results show how important it is to add these realistic factors to evaluate the performance of a dynamic production grid. Disturbances have a large impact on the efficiency of the grid. However, queue jumping and rescheduling behaviour show that these problems can be effectively mitigated.

What can be concluded is that manufacturing systems using agent-based technology can anticipate disturbances and reconfigure systems on demand. Which brings dynamic automated production for high-mix, low-volume closer to be a realistic business for industry.

8.5 Future Work

The current simulations have been used to show how Grid Manufacturing could possibly be used and organised. It shows the possible effects in realistic proof of concept cases and how to counteract them. However, many opportunities are still open to be explored in the future, among them:

- The parameters used in current cases are only used to illustrate the current effects and use of a grid of equiplets. However, they might be optimised and changed to encompass a full range of different cases and validate the real economical efficiency of grids. Since this research is currently mainly focused on the technical capabilities of Grid Manufacturing this is considered to be out of scope. However, in the future

it could be interesting to use similar simulation to explore the matter further.

- The simulation could also be integrated into the REXOS system, much like the Gazebosim has been integrated, which could give a number of opportunities:
  - To use statistical data to create a prediction when an equiplet should be reconfigured. This could be done by analysing current and expected demand based on data analytics and give a precise estimation when the cost of reconfiguring an equiplet would be efficient.
  - The simulation could automatically chose management strategies and optimise schedules for both products and equiplets.
- Scheduling is mainly placed out of scope for this research, since, while important, it does not have a fundamental effect on the technical capabilities of the grid. However, this research and the simulator that was developed can be used to quickly analyse scheduling efficiency and improve Grid Manufacturing even more.

8.6 Conclusion

The main Research questions for the current chapter were:

RQ5a - How can we validate the system in different cases? A simulation can be used that emulate a diversity of cases. In the current simulation a number of possibilities are validated that Grid Manufacturing provides, including reconfiguration, a mix of organised control with a hierarchy or heterarchy, and the ability to adapt to disturbances in the manufacturing process.

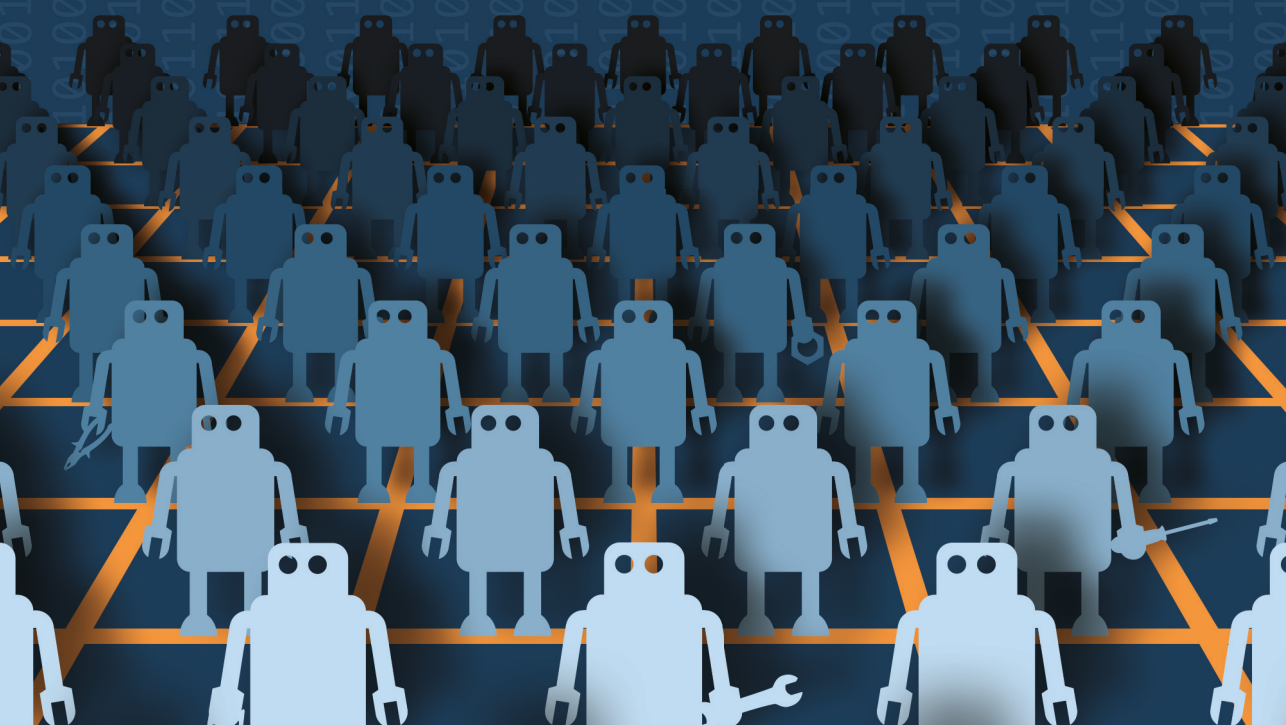
RQ5b - What are the unique factors that influence the production efficiency of a grid? Reconfiguration and disturbances like breakdowns are important factors. The reactive simulations show that Grid Manufacturing is efficient when high-mix, low-volume products are produced. However, the dynamic behaviour does greatly impact the efficiency. This can be mitigated by introducing new strategies for scheduling and adapting to the circumstances.

RQ5c - What strategies or optimisations can counter the expected negative factors like disturbances in the manufacturing process? Queue jumping and rescheduling were introduced to counter the negative effects. These proved to mitigate the problems efficiently.

RQ5d What management strategies can be used to control a grid during various cases? Grid Manufacturing was designed to be a completely heterarchical system that was self-managed by the product and equiplet agents.

However, flexibility could always be limited on demand, by creating a less flexible system with hierarchical optimisations. In case of larger batches, or different priorities for the production this could be used to adapt the Grid for a specific purpose. While not designed for mass production of similar products, equiplets could be used as a classic manufacturing line. Using the grid this way would limit some of its flexibility, but with their low-cost design and flexible setup it could also be an efficient way to quickly install a new manufacturing line for larger batch production.

09



Discussion

“We are trying to prove ourselves wrong as quickly as possible, because only in that way can we find progress.”

— Richard Feynman

Discussion

9.1 General Evaluation

This entire study has been about the technical possibilities for automated manufacturing of differentiated products, by using the concept of Grid Manufacturing. This starts in the first main Chapter 4, by investigating how to flexibly localise and handle objects. Next reconfiguration is discussed in Chapter 5. This adds ways to change hardware and discusses how products make use of these new services. This brings flexibility not only in how to use the equiplets, but even to the hardware configuration itself by using the Hardware Abstraction Layer and creating a data-driven system that could adapt at any time and on demand. Using the generic methods of these two chapters the architecture has been founded in Chapter 6. While this chapter was mainly about development, between the lines some flexibility could be found, for instance 'product family engineering', which makes it easier to reuse objects, and how to use hybrid architecture to combine both performance and more intelligent systems. Besides Multi-Agent Systems, it also introduced Robot Operating System, a middleware that makes use of autonomous nodes. When the architecture was completed it was time to investigate essential aspects of the systems to make it possible to use it in industry. This was done in Chapter 7 that showed a standardised state system and simulators that could predict collisions before they can happen. Finally Chapter 8 zoomed out on the entire grid, making use of all the systems and tested them using simulations, including logistics, new ways to organise production and strategies to use reconfigurable systems.

While all closely connected, these subjects come from a diverse range: from hardware control to multi-agent systems and logistic control problems. However, in a similar sense as Leitão (2009) discusses in his survey, to prove this technology to industry, it is essential to consider practical problems together with the fundamental ones. This is the reason why this study includes elements like safety and an experimental platform. These make it possible to test the more generalisable elements, like the 6D localisation approach and the Hardware Abstraction Layer approach for reconfiguration. However, the requirement for a working implementation introduces a broad spectrum of practical and multi-disciplinary problems. As such, substantial energy has been put into pragmatic problem solving. This is a common challenge for design research, as Horvath et al. (2013) also mentions. It is difficult to achieve the rigour of the 'hard' sciences in this field. However, Horvath did mention

possibilities to articulate and increase the methodological structure. In retrospect to this thesis it can be said that the applied and multi-disciplinary approach did provide extra challenges. These challenges create a difficult balance between showing the practical 'working' methods on one hand and the possibility to limit the scope to create a more rigorous model on the other hand. The goal was to make applicable systems for industry. Hence, at times it was chosen to take a pragmatic approach that would create a working system that can be used for future experimentation. In retrospect this meant that at times the choice was made to make a more complex applied system that was usable instead of creating a simpler and generalisable model that would not be applicable in the near future. Since several working proofs of concept are now available, this can be used in the future for further development, research, analyses and generalisation.

9.2 Review of the Research Design Parameters

RP1. Machines can effectively provide generic services that can be used by different kind of products that are not known a priori, i.e. new products can be built on demand by machines never specifically designed for this product. This objective was reached by creating the Hardware Abstraction Layer infrastructure, which was discussed in Chapter 5. The HAL makes this possible by using the Knowledge Database to translate the abstract standardised 'product steps' to specific instructions for the hardware configuration that has been scheduled by the Product Agent.

One of the HAL's drawbacks is that it is a complex system with an implementation of more than 25 software classes. However, since it holds an important functionality this seems unavoidable at the moment. Nevertheless, what could be done is develop a number of tools that simplify the use of the HAL and improves the documentation of how it can be used and maintained. This could subsequently improve maturity and simplify its use. However, since this is a more practical solution than a fundamental one this was given a lower priority for this study.

RP2. Computer vision is an important part of flexible manufacturing and can be simplified by making use of diverse data that is already available in the system. This was shown in Chapter 4. Machine Vision in general is a complicated problem that commonly has to be tailored for every specific application. However, this problem is greatly simplified by making use of the models and information available with the Cyber-Physical approach. Just two 2D cameras can now accurately determine the positioning of Objects.

While the proposed system is rather elegant, it does only work with objects

that have a model. Other problems like unknown objects (humans nearby the machines, for instance) are not taken into account. In future studies this could be added to not only identify known models, but also recognise 'unknown' objects. While these cannot be localised with this system it could provide more safety and give information about the environment. The current system also has limitations, which were discussed in more detail in Chapter 4.

RP3. The hardware of a machine can be reconfigured without the need of reprogramming the software. This was shown in Chapter 5. The Hardware Abstraction Layer has a number of factories that can create capabilities and modules in a way that is purely 'data-driven'. By this design any new system can be added to the configuration purely based on the abilities, i.e. mutations, that the module can perform. Together with the ability to start and stop a modules software on demand with the use of ROS nodes, as shown in Chapter 6, a system is created where hardware modules can be reconfigured without the need for reprogramming.

Considering RP3, what could be improved for the Hardware Abstraction Layer, is the way it can be adapted. In the future a tool should be made that can easily access the Knowledge Database and can add or remove newly developed modules or product steps. This is one of the drawbacks of the current implementation. While the back-end (the HAL) itself is mature, the front-end is not. It could be possible to make a Graphical User Interface that has a good User Experience (UX) design, e.g. such that the reconfiguration process itself would be nothing more than just scanning a barcode. The GUI could also provide means to change variables into the Knowledge Database and give more status information about the configuration. At the moment these values are hidden into the back-end, making it more difficult to change details or test new modules.

RP4. The use of a simulator and transparent software control using standardised states can increase safety for reconfigurable manufacturing machines. Chapter 7 discusses the risks that are introduced by the use of reconfigurable manufacturing machines in combination with the uncertainties of handling a priori unknown products. These risks have to be mitigated. The use of a simulator and the state/mode system that define behaviour and predict possible problems does increase safety.

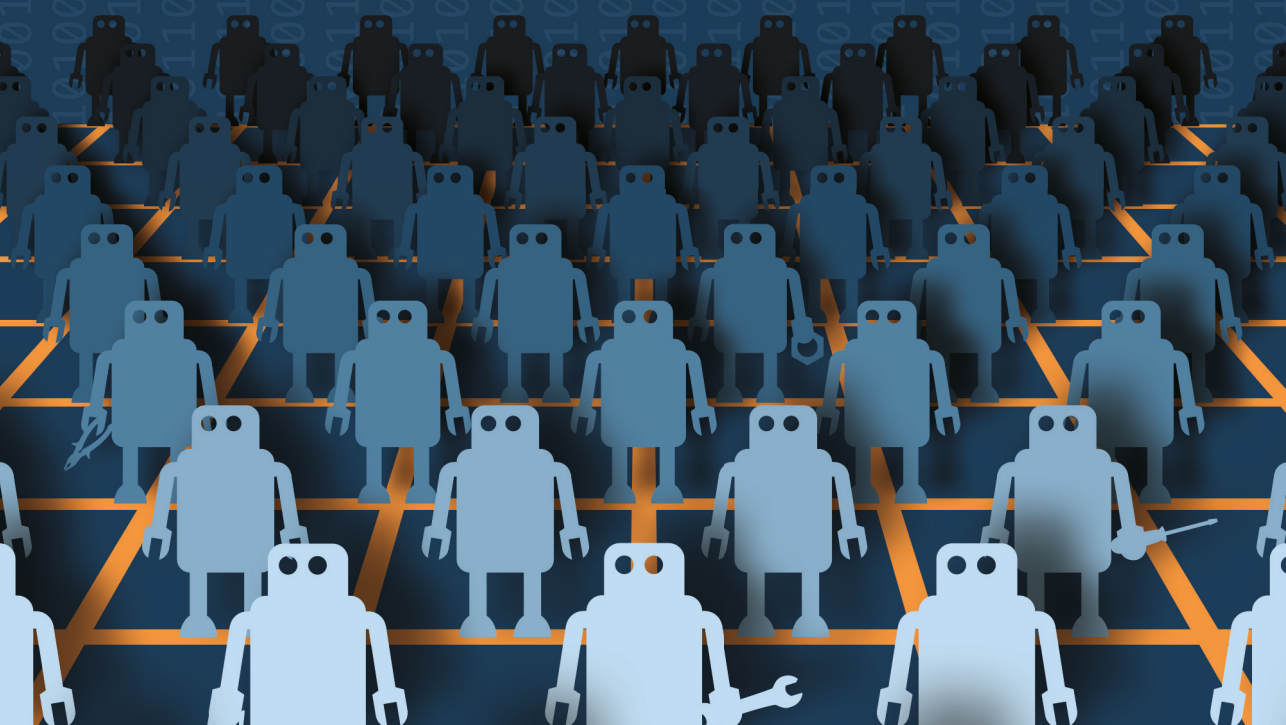
While the simulator and software control system do increase safety, they do increase complexity and cost to the development of new reconfigurable modules. As discussed in RP4 reconfigurability, reprogramming is not required for the reconfiguration process itself. However, the development of new hardware modules do require more work to be developed. An accurate dynamic model has to be developed that can be used for the simulation and there has to be a clear definition of what a system is allowed to do within a state. Hence, the development of new reconfigurable modules are more complex than with

classic manufacturing means and therefore new hardware modules has to be tested thoroughly to ensure safe operation.

RP5. Using cooperating agents in a grid in a non-hierarchical, i.e. heterarchical manner can be a flexible and efficient way to manufacture products in low quantities. Chapter 8 introduced new ways to organise a grid-like system and tests these by using various self-developed simulators. This is one of the strong points of autonomous systems, which might be designed in a heterarchical fashion, but can easily be changed towards a hierarchical efficient line or any other hybrid variation. Adapting towards the current needs is therefore the key to make efficient use of these systems.

While several cases were discussed for RP4, this does not nearly cover all opportunities. However, several dissertations would be required to research all possibilities that exist for organisation modelling, especially if topics like scheduling, transport logistics and optimisation strategies for reconfiguration are taken into account. Also the efficiency, business strategies and cost studies could be added here. However, this was placed outside of the scope of this thesis. Here we just introduce the technical possibilities and ideas of how to utilise them, in the hope they will be picked up for further studies in multidisciplinary research groups and industry.

10



Conclusion

“There’s nothing I believe
in more strongly than
getting young people
interested in science and
engineering, for a better
tomorrow, for all humankind”

—Bill Nye

Conclusion

10.1 Review of the Research Questions

In this section the research questions will be reviewed:

RQ1 - How can the detection and localisation of (previously unknown) objects be simplified and generalised? Simplification of the (6D) localisation problem has been done by using metadata that is available in the grid. The use of the Cyber-Physical philosophy creates unique opportunities to do this. Since all systems in Grid Manufacturing have knowledge of themselves, including their state, configuration, and even visual CAD models, they are able to provide rich data that increases accuracy and that can be used to verify the results. For this purpose several tools were created that can automatically generate images for matching purposes on demand, creating a flexible approach to be able to identify and localise any product (or other object) that has this data available. See Chapter 4 for more details.

RQ2 - Can reconfigurable systems be controlled without the need to reprogram them for every new product or hardware module? This can be done by making the system Data-Driven and using an architecture where new entities can be started and stopped on demand. The combination of these two elements are essential to do this. In this study we use the Multi-Agent Platform JADE, and Robot Operating Systems which both are able to do this. However, this involves numerous systems, the platforms, but also a translation system so that the service can be used by products, and the Hardware Abstraction Layer that is used to perform the configuration and translation process by using the Knowledge Database. See Chapter 5 and 6 for more details.

RQ3 - What options are available to combine flexibility and performance for software architecture in grid manufacturing? As discussed in Chapter 6, flexibility and performance are often not easy to combine. However, Axiomatic Design dictates that when requirements conflict, they should be decoupled. This is done by using a Hybrid Architecture that uses two different platforms that each have their own strong points. Of course the question then arises on how these platforms are defined and if this impacts performance. These aspects have been discussed in more detail in this chapter.

RQ4 - What Risks are introduced due to the reconfigurable and dynamic behaviour and how can they be mitigated? Grid Manufacturing is supposed to be agile, but agile also means volatile in this aspect. Different configurations, different products and a continuously changing work-

ing area make this a serious problem. However, this has been mitigated in two manners. First by creating a standardised state system that provides clear insight in the behaviour of the system. Secondly, the idea of Cyber-Physical system is also applied by using the 'cyber' world to test actions in the virtual world, before actually attempting them in the real 'physical' world. The combination of these two make the system safer to use and lowers the risk that come inherently with dynamic agile systems. See Chapter 7 for more details.

RQ5 - What is the best way to utilise the possibilities of an agent-based manufacturing grid and therewith validate its efficiency? This Research question is the hardest to answer, since efficiency is dependent on the case and the current Grid Manufacturing proof of concept is designed especially for automated flexibility, not to determine what the most optimal utilisation would be. Since the current proof of concept is tailored on flexibility, it could be approached in many ways. However, Chapter 8 does create several simulations to test the efficiency and experiment with new possibilities. So in general it can be stated that Grid Manufacturing as a concept provides new organisational models, and lowers the time to market. Even the automated manufacturing of one prototype product, without an impact into an active production systems, is now easier to achieve. Any product can enter the grid at any time, as long as the parts and capabilities to assemble them are available. In general the concept could provide opportunities to make high-mix, low-volume production more cost-efficient for automated manufacturing.

This brings us to the main research question. **RQ0 - What could be the role of Reconfigurable Manufacturing Machines in the automation of high-mix, low-volume production?** To enable flexibility and be able to perform automated high-mix, low-volume production were the most important aspects of this research. Reconfigurable Manufacturing Systems were an essential and central parts in this. However, the equiplets used in Grid Manufacturing expands the context of reconfiguration much further than just being able to change the hardware. Grid Manufacturing encompasses the three main key interests, as mentioned in Chapter 1 Introduction: (1) Reconfigurability; (2) Lowering Complexity; and (3) Autonomy. A manufacturing grid in the current form is therefore an automated system that can adapt its hardware, capabilities and immediately offer its new service to products. An important aspect to be able to do this is through the use of Cyber-Physical Systems. The integration of the Cyber (software) and Physical (control software) perspective brings many opportunities. Examples for this are are given all throughout the thesis, e.g. in Chapter 4 in the context of object awareness, where object models and knowledge is combined with the configuration knowledge of the equiplet agent to confirm the location of a product.

To come back to the main question. Without Reconfigurable Manufacturing Systems the possibilities to automate high-mix low volume products would

be limited. However, as also noted in the IBM whitepaper about autonomic computing, the automation of (IT) infrastructure is an evolutionary process (Computing, 2003). This also fits into the concept of Grid Manufacturing, as seen from the perspective of the four categories mentioned in autonomic computing, i.e.:

- Self-configuration - as mentioned in Chapter 5, systems can reconfigure and automatically offer new services. Chapter 8 takes this even further on a larger scale, by showing how reconfiguration can be applied on a larger grid scale to adapt to new demand.
- Self-healing - as shown in Chapter 8, the grid can detect disturbances and react to this by either replanning or reconfiguring such that a product can still be manufactured.
- Self-optimising - Chapter 8 shows how RMS can be used to optimise demand, by monitoring overall equiplet utilisation and reconfigure equiplets to a capability with a higher demand.
- Self-protecting - In the context of Grid Manufacturing this can be seen as the way that a system can protect itself from harm. As was described in Chapter 7, treats like collisions or dangerous moves can now be avoided.

Reconfiguration plays an important part seen from all these four categories. Hence, reconfigurability, and therefore RMS, plays an essential role in the automation of high-mix, low-volume production.

10.2 Final Conclusion

Grid Manufacturing involves a great number of fields, this study looks at the technical aspects and has found a number of opportunities. While Grid Manufacturing in general requires more research to become fully mature for industry, other aspects have been identified that could be used in the short term. Hence, the study in general can be seen as another step into the introduction of more Cyber-Physical Systems in industry. These are eight of the achievements that have been made throughout this PhD study:

1. The 6D localisation process - based on several (standalone) tools, software engineering, and computer vision, is able to identify and localise objects in 6D, using two 2D cameras and CAD models;
2. The translation process for products to use generic services - Based on the ability to translate abstract product steps into specific hardware operations for any known configuration;

3. The Hardware Abstraction Layer - which is able to change the configuration of machines by starting/stopping/changing new software entities by using data-driven systems;

4. A Hybrid Architecture - which combines flexibility and performance by tying together a Multi-Agent System and the Hardware controlling Robot Operating System, connecting by a high-performance middleware system;

5. A standardised State System - which is tailored for reconfigurable and autonomous manufacturing machines to give more insight in their current status for safety purposes.

6. A simulator for reconfigurable manufacturing machines - which can dynamically test new configurations or manufacturing processes and also determine if movements can be safely performed.

7. New organisational simulators - which test new organisation models, i.e. if autonomous machines that are self manageable should actually be self-managed or perhaps could better be controlled in a hierarchical manner.

8. Solutions to practical problems - Straightforward ways like queue jumping and reconfiguration that adapt to practical problems like disturbances and overloaded equiplets that will likely occur in a dynamic production system like Grid Manufacturing.

These achievements provide opportunities for more flexibility in manufacturing, potentially opening new business opportunities and efficiency in the manufacturing industry.

References

Akhras, G. (1997). Smart structures and their, applications in civil engineering. *Civil, Engineering Report, CE97-2, RMC, Kingston, Ontario, Canada*.

Applegate, D. and Cook, W. (1991). A computational study of the job-shop scheduling problem. *OSRA Jounal of Computing*, 3(2):149–156.

Arkin, R. C. (1998). *Behavior-based robotics*. MIT press.

Azad, P., Asfour, T., and Dillmann, R. (2007). Stereo-based 6d object localization for grasping with humanoid robot systems. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 919–924.

Bakker, T. and Telgen, D. (2015). T-rex, a simulation for reconfigurable manufacturing machines. unpublished technical document.

Barata, J., Camarinha-Matos, L., and Cndido, G. (2008). A multiagent-based control system applied to an educational shop floor. *Robotics and Computer-Integrated Manufacturing*, 24(5):597 – 605. {BASYS} '06: Balanced Automation Systems7th {IFIP} International Conference on Information Technology for Balanced Automation Systems in Manufacturing and Services.

Barbosa, J. and Leitao, P. (2011). Simulation of multi-agent manufacturing systems using agent-based modelling platforms. In *Industrial Informatics (INDIN), 2011 9th IEEE International Conference on*, pages 477–482.

Bellifemine, F., G., C., and Greenwood, D. (2007). *Developing multi-agent systems with Jade*. John Wiley & Sons Ltd.

Bourne, D. A. and Fox, M. S. (1984). Autonomous manufacturing: automating the job-shop. *Computer*, 17(9):76–86.

Bratman, M. (1987). *Intention, Plans, and Practical Reason*. Harvard University Press, Cambridge, Mass.

Braubach, L., Pohkahr, A., and Lamersdorf, W. (2004). Jadex: A short overview. *Proceedings of the Main Conference Net.ObjectDays*.

Brun, A. and Portioli, A. (1999). Agent-based shop-floor scheduling of multi stage systems. *Computers & Industrial Engineering*, 37(12):457 – 460. Proceedings of the 24th international conference on computers and industrial engineering.

Cho, S. and Prabhu, V. V. (2007). Distributed adaptive control of production scheduling and machine capacity. *Journal of Manufacturing Systems*, 26(2):65 – 74. Distributed Control of Manufacturing SystemsSpecial Issue: Distributed Control of Manufacturing Systems.

Computing, A. (2003). An architectural blueprint for autonomic computing. *IBM Publication*.

Dastani, M. (2008). 2apl: a practical agent programming language. *Autonomous Agents and Multi-Agent Systems*, 16(3):214–248.

De Koster, R., Le-Duc, T., and Roodbergen, K. J. (2007). Design and control of warehouse order picking: A literature review. *European Journal of Operational Research*, 182(2):481–501.

Dennett, D. (1987). *The Intentional Stance*. MIT Press, Cambridge, Mass.

Dilts, D., Boyd, N., and Whorms, H. (1991). The evolution of control architectures for automated manufacturing systems. *Journal of Manufacturing Systems*, 10(1):79 – 93.

Duffie, N. A. and Prabhu, V. V. (1994). Real-time distributed scheduling of heterarchical manufacturing systems. *Journal of Manufacturing Systems*, 13(2):94 – 107.

ElMaraghy, H. (2005). Flexible and reconfigurable manufacturing systems paradigms. *International Journal of Flexible Manufacturing Systems*, 17(4):261–276.

Endsley, E., Almeida, E., and Tilbury, D. (2006). Modular finite state machines: Development and application to reconfigurable manufacturing cell controller generation. *Control Engineering Practice*, 14(10):1127 – 1142. The Seventh Workshop On Discrete Event Systems (WODES2004)The Seventh Workshop On Discrete Event Systems (WODES2004).

Flynn, P. J. and Jain, A. K. (1989). Cad-based computer vision: from cad models to relational graphs. In *Systems, Man and Cybernetics, 1989. Conference Proceedings., IEEE International Conference on*, pages 162–167 vol.1.

Gibson, J. (1977). *The Theory of Affordances*, pages 67–82. Hillsdale.

Giret, A. and Botti, V. (2009). Engineering holonic manufacturing systems. *Computers in Industry*, 60(6):428 – 440. Collaborative Engineering: from Concurrent Engineering to Enterprise Collaboration.

Gunasekaran, A. (1999). Agile manufacturing: a framework for research and development. *International journal of production economics*, 62(1):87–105.

Hall, K. H., Staron, R. J., and Vrba, P. (2005). Experience with holonic and agent-based control systems and their adoption by industry. In Mak, V., William Brennan, R., and Pchouek, M., editors, *Holonic and Multi-Agent Systems for Manufacturing*, volume 3593 of *Lecture Notes in Computer Science*, pages 1–10. Springer Berlin Heidelberg.

Heintz, F., Rudol, P., and Doherty, P. (2007). Bridging the sense-reasoning gap using dyknow: A knowledge processing middleware framework. In Hertzberg, J., Beetz, M., and Englert, R., editors, *KI 2007: Advances in Artificial Intelligence*, volume 4667 of *Lecture Notes in Computer Science*, pages 460–463. Springer Berlin Heidelberg.

Höpf, M. and Schaeffer, C. (1997). *Information Infrastructure Systems for Manufacturing: Proceedings of the IFIP TC5/WG5.3/WG5.7 international conference on the Design of Information Infrastructure Systems for Manufacturing, DIISM '96 Eindhoven, the Netherlands, 15–18 September 1996*, chapter Holonic Manufacturing Systems, pages 431–438. Springer US, Boston, MA.

Horvath, I. et al. (2013). Structuring the process of design research-a necessary step towards ensuring scientific rigor. In *Proceedings of the 19th international conference on engineering design*, pages 1–10.

HU, S. (2005). Paradigms of manufacturing - a panel discussion. *3rd Conference on Reconfigurable Manufacturing*.

Järvenpää, E., Luostarin, P., Lanz, M., and Tuokko, R. (2012). Development of a rule-base for matching product requirements against resource capabilities in an adaptive production system. *FAIM2012 proceedings, Helsinki*, pages 449–456.

Jennings, N. R. (2001). An agent-based approach for building complex software systems. *Commun. ACM*, 44(4):35–41.

Kallmann, M. and Thalmann, D. (1999). Modeling objects for interaction tasks. In Arnaldi, B. and Hgron, G., editors, *Computer Animation and Simulation 98*, Eurographics, pages 73–86. Springer Vienna.

Kelton, W. D. and Law, A. M. (2000). *Simulation modeling and analysis*. McGraw Hill Boston.

Khalgui, M. and Mosbahi, O. (2010). Intelligent distributed control systems. *Information and Software Technology*, 52(12):1259 – 1271.

Koh, S. L. and Wang, L. (2010). Overview of enterprise networks and logistics for agile manufacturing. In Wang, L. and Koh, S. L., editors, *Enterprise Networks and Logistics for Agile Manufacturing*, pages 1–10. Springer London.

Komma, V., Jain, P., and Mehta, N. (2011). An approach for agent modeling in manufacturing on jade reactive architecture. *The International Journal of Advanced Manufacturing Technology*, 52(9-12):1079–1090.

Koren, Y. (2006). General rms characteristics. comparison with dedicated and flexible systems. In *Reconfigurable manufacturing systems and transformable factories*, pages 27–45. Springer.

Koren, Y. and Shpitalni, M. (2010). Design of reconfigurable manufacturing systems. *Journal of Manufacturing Systems*, 29(4):130 – 141.

Kortuem, G., Kawsar, F., Fitton, D., and Sundramoorthy, V. (2010). Smart objects as building blocks for the internet of things. *Internet Computing, IEEE*, 14(1):44–51.

Laloux, F. (2015). *Reinventing organizations*. Lannoo Meulenhoff-Belgium.

Lee, E. A. and Seshia, S. A. (2011). *Introduction to embedded systems: A cyber-physical systems approach*. Lee & Seshia.

Lee, J., Bagheri, B., and Kao, H.-A. (2015). A cyber-physical systems architecture for industry 4.0-based manufacturing systems. *Manufacturing Letters*, 3:18 – 23.

Leitão, P. (2009). Agent-based distributed manufacturing control: A state-of-the-art survey. *Eng. Appl. Artif. Intell.*, 22(7):979–991.

Lowe, D. G. (1987). Three-dimensional object recognition from single two-dimensional images. *Artificial Intelligence*, 31(3):355 – 395.

Marik, V. and McFarlane, D. (2005). Industrial adoption of agent-based technologies. *IEEE Intelligent Systems*, 20(1):27–35.

Maturana, F., Staron, R., Hall, K., Tich, P., lechta, P., and Mak, V. (2005). An intelligent agent validation architecture for distributed manufacturing organizations. In Camarinha-Matos, L., editor, *Emerging Solutions for Future Manufacturing Systems*, volume 159 of *IFIP International Federation for Information Processing*, pages 81–90. Springer US.

McFarlane, D. and Bussmann, S. (2003). Holonic manufacturing control: Rationales, developments and open issues. In Deen, S., editor, *Agent-Based*

Manufacturing, Advanced Information Processing, pages 303–326. Springer Berlin Heidelberg.

Mcfarlane, D. C. and Bussmann, S. (2000). Developments in holonic production planning and control. *Production Planning & Control*, 11(6):522–536.

Mehrabi, M., Ulsoy, A., and Koren, Y. (2000). Reconfigurable manufacturing systems: Key to future manufacturing. *Journal of Intelligent Manufacturing*, 11(4):403–419.

Moergestel, L. v. (2014). *Agent Technology in Agile Multiparallel Manufacturing and Product Support*. PhD thesis Utrecht University, Utrecht, the Netherlands.

Moergestel, L. v., Meyer, J.-J., Puik, E., and Telgen, D. (2010). The role of agents in the lifecycle of a product. *CMD 2010 proceedings*, pages 28–32.

Moergestel, L. v., Meyer, J.-J., Puik, E., and Telgen, D. (2011). Decentralized autonomous-agent-based infrastructure for agile multiparallel manufacturing. *Proceedings of the International Symposium on Autonomous Distributed Systems (ISADS 2011) Kobe, Japan*, pages 281–288.

Moergestel, L. v., Meyer, J.-J., Puik, E., and Telgen, D. (2012). Production scheduling in an agile agent-based production grid. *Proceedings of the Intelligent Agent Technology (IAT 2012)*, pages 293–298.

Moergestel, L. v., Meyer, J.-J., Puik, E., and Telgen, D. (2013a). Embedded autonomous agents in products supporting repair and recycling. *Proceedings of the International Symposium on Autonomous Distributed Systems (ISADS 2013) Mexico City*, pages 67–74.

Moergestel, L. v., Meyer, J.-J., Puik, E., and Telgen, D. (2013b). A versatile agile agent-based infrastructure for hybrid production environments. *IFAC Modeling in Manufacturing proceedings, Saint Petersburg*, pages 210–215.

Moergestel, L. v., Meyer, J.-J., Puik, E., and Telgen, D. (2014). Agent-based manufacturing in a production grid: Adapting a production grid to the production paths. *Proceedings of the International Conference on Agents and Artificial Intelligence (ICAART 2014)*, 1:342–349.

Moergestel, L. v., Puik, E., Telgen, D., Meyer, J.-J., et al. (2015). A study on transport and load in a grid-based manufacturing system. *International Journal on Advances in Software*, 8(1 & 2):27–37.

Moore, G. A. (1998). *Inside the tornado*. Capstone.

Murphy, L. M. and Edwards, P. L. (2003). *Bridging the valley of death: Transitioning from public to private sector financing*. National Renewable Energy Laboratory Golden, CO.

Olhager, J. (2010). The role of the customer order decoupling point in production and supply chain management. *Computers in Industry*, 61(9):863–868.

Onori, M. and Barata Oliveira, J. (2010). Outlook report on the future of european assembly automation. *Assembly Automation*, 30(1):7–31.

Paolucci, M. and Sacile, R. (2005). *Agent-based manufacturing and control systems : new agile manufacturing solutions for achieving peak performance*. CRC Press, Boca Raton, Fla.

Porter, M. E. and Advantage, C. (1985). Creating and sustaining superior performance. *Competitive advantage*, page 167.

Puik, E. (2016). *Risk Adjusted, Concurrent Development of Microsystems and Reconfigurable Manufacturing Systems*. PhD thesis manuscript University of Warwick.

Puik, E., Gerritsen, J., Smulders, E., van Huijgevoort, B., and Ceglarek, D. (2013a). A method for indexing axiomatic independence applied to reconfigurable manufacturing systems. In *7th International Conference on Axiomatic Design ICAD, 1st edn., Worcester*, volume 7, pages 186–194.

Puik, E. and Moergestel, L. (2010). Agile multi-parallel micro manufacturing using a grid of equiplets. In Ratchev, S., editor, *Precision Assembly Technologies and Systems*, volume 315 of *IFIP Advances in Information and Communication Technology*, pages 271–282. Springer Berlin Heidelberg.

Puik, E., Telgen, D., Moergestel, L., and Ceglarek, D. (2013b). Structured analysis of reconfigurable manufacturing systems. In Azevedo, A., editor, *Advances in Sustainable and Competitive Manufacturing Systems*, Lecture Notes in Mechanical Engineering, pages 147–157. Springer International Publishing.

Quigley, M., Conley, K., Gerkey, B. P., Faust, J., Foote, T., Leibs, J., Wheeler, R., and Ng, A. Y. (2009). Ros: an open-source robot operating system. In *ICRA Workshop on Open Source Software*.

Rao, A. S., Georgeff, M. P., et al. (1995). Bdi agents: From theory to practice. In *ICMAS*, volume 95, pages 312–319.

Rauschecker, U., Stock, D., Stöhr, M., and Verl, A. (2014). Connecting factories and related it environments to manufacturing clouds. *International Journal of Manufacturing Research*, 9(4):389–407.

Ricci, A., Viroli, M., and Piunti, M. (2010). Formalising the environment in mas programming: A formal model for artifact-based environments. In Braubach, L., Briot, J.-P., and Thangarajah, J., editors, *Programming Multi-Agent Systems*, volume 5919 of *Lecture Notes in Computer Science*, pages 133–150. Springer Berlin Heidelberg.

Rockett, P. (1999). The accuracy of sub-pixel localisation in the canny edge detector. In *BMVC*, pages 1–10. Citeseer.

Schild, K. and Bussmann, S. (2007). Self-organization in manufacturing operations. *Commun. ACM*, 50(12):74–79.

Shen, W., Hao, Q., Yoon, H. J., and Norrie, D. H. (2006). Applications of agent-based systems in intelligent manufacturing: An updated review. *Advanced engineering INFORMATICS*, 20(4):415–431.

Stappen, F. v. d., Berretty, R.-P., Goldberg, K., and Overmars, M. H. (2002). Geometry and part feeding. In *Sensor Based Intelligent Robots*, pages 259–281. Springer.

Stecke, K. (2005). Flexibility is the future of reconfigurability. paradigms of manufacturinga panel discussion. In *3rd Conference on Reconfigurable Manufacturing*.

Suh, N. P. (1999). A theory of complexity, periodicity and the design axioms. *Research in Engineering Design*, 11(2):116–132.

Suh, N. P. (2001). *Axiomatic Design: Advances and Applications (The Oxford Series on Advanced Manufacturing)*. Oxford University Press, USA.

Telgen, D., Brink, L. v. d., Moergestel, L. v., Puik, E., and Meyer, J.-J. (2015a). Adding reconfiguration to an agile agent based production grid. In *Proceedings of the 25th International Conference on Flexible Automation and Intelligent Manufacturing - Designing for Advanced, High Value Manufacturing and Intelligent Systems for the 21st Century - Volume I*, pages 496–503. The Choir Press.

Telgen, D., Moergestel, L. v., Puik, E., Streng, A., Scheefhals, R., Bakker, T., Hustinx, A., Brink, L. v. d., and Meyer, J.-J. (2014). Hierarchical management of a heterarchical manufacturing grid. In *Proceedings of the 24th International Conference on Flexible Automation and Intelligent Manufacturing FAIM 2014*, pages 825–832. DEStech Publications, Inc.

Telgen, D., Puik, E., van Moergestel, Tommas, B., and Meyer, J.-J. (2015b). Reconfigurable equiplets operating system - a hybrid architecture to com-

bine flexibility and performance for manufacturing. *International Journal On Advances in Software*, 8(3 & 4):309 326.

Telgen, D., Puik, E., van Moergestel, L., and Meyer, J.-J. (2013a). Distributed and heterarchical control for grid manufacturing. In *International conference on Instrumentation, Control, Information Technology and System Integration, Proceedings of*, pages 526–531. IEEE.

Telgen, D., van Moergestel, L., Puik, E., and Meyer, J.-J. (2012). Agile manufacturing possibilities with agent technology. In *Proceedings of 22nd International Conference on Flexible Automation and Intelligent Manufacturing*, pages 341–346. Tampere University of Technology.

Telgen, D., van Moergestel, L., Puik, E., Muller, P., Groenewegen, A., van der Steen, D., Koole, D., de Wit, P., van Zanten, A., and Meyer, J.-J. (2013b). Combining performance and flexibility for rms with a hybrid architecture. In *Advances in Artificial Intelligence, 16th Portuguese Conference on Artificial Intelligence, local proceedings*, pages 388–399. IEEE.

Telgen, D., van Moergestel, L., Puik, E., Muller, P., and Meyer, J.-J. (2013c). Requirements and matching software technologies for sustainable and agile manufacturing systems. In *INTELLI 2013, The Second International Conference on Intelligent Systems and Applications*, pages 30–35.

Trentesaux, D. (2009). Distributed control of production systems. *Engineering Applications of Artificial Intelligence*, 22(7):971 – 978. Distributed Control of Production Systems.

van den Brink, L. (2015). Emulation of an agent-based reconfigurable manufacturing grid. Master Thesis Laurens van den Brink, supervised by Daniel Telgen and John-Jules Meyer.

van Moergestel, L., Puik, E., Telgen, D., Kuijl, M., Alblas, B., Koelewijn, J., Meyer, J.-J., et al. (2014). A simulation model for transport in a grid-based manufacturing system. In *Proc. of the Third International Conference on Intelligent Systems and Applications (INTELLI 2014)*, pages 1–7. IARIA.

van Moergestel, L., Puik, E., Telgen, D., and Meyer, J.-J. (2011). Decentralized autonomous-agent-based infrastructure for agile multiparallel manufacturing. In *Autonomous Decentralized Systems (ISADS), 2011 10th International Symposium on*, pages 281–288. IEEE.

van Moergestel, L., Puik, E., Telgen, D., and Meyer, J.-J. (2012). Production scheduling in an agile agent-based production grid. In *Web Intelligence and Intelligent Agent Technology (WI-IAT), 2012 IEEE/WIC/ACM International Conferences on*, volume 2, pages 293–298.

Wang, L., Song, Y., and Gao, Q. (2009). Designing function blocks for distributed process planning and adaptive control. *Engineering Applications of Artificial Intelligence*, 22(7):1127 – 1138. Distributed Control of Production Systems.

Weerdt, M. d. and Clement, B. (2009). Introduction to planning in multiagent systems (preprint). *Multiagent and Grid Systems An International Journal*, 5(4):345–355.

Wind, J. and Rangaswamy, A. (2001). Customerization: The next revolution in mass customization. *Journal of Interactive Marketing*, 15(1):13 – 32.

Wooldridge, M. (2009). *An Introduction to MultiAgent Systems, Second Edition*. Wiley, Sussex, UK.

Wooldridge, M. and Jennings, N. (1995). Intelligent agents: Theory and practice. *The Knowledge Engineering Review*, 10:115–152.

Abstract

Each period of time has its own revolution, and with each revolution comes its own organisational model. We find ourselves in the 4th industrial revolution, where the internet of things connects autonomous embedded systems that live both in the virtual 'cyber' world, and in the real 'physical' world. These so-called Cyber-Physical Systems follow modern organisational models like self-management and can take proactive actions themselves.

The thesis zooms in from the Cyber-Physical perspective to manufacturing systems that are both reconfigurable, autonomous, and extremely flexible. This can only be achieved by developing new methods and using technologies that enable flexibility. However, efficiency has to be improved as well, e.g. by making assembly so flexible that it becomes cost-efficient to automate the production of low quantities of different products: so-called high-mix, low-volume production. The ability to automatically manufacture high-mix, low-volume production will drive the market for mass customisation, and bring about a shorter time to market. In practice, the move for flexibility will be achieved by creating new methods and tools, combining new technologies, and applied testing with simulators and newly-developed manufacturing systems.

This thesis start by introducing the concept behind the manufacturing methodology, called 'Grid Manufacturing'. Grid Manufacturing takes place using autonomous agents for both the equipment and the product itself. Products live in the 'cyber' world even before they are created in the 'physical' world and are aware of how they should be manufactured. They communicate and negotiate with reconfigurable manufacturing machines, called 'equiplets'. The equiplets offer generic services, e.g. pick & place, which the products can use on demand by negotiating a place into the equiplets schedule. This study focuses on the design and technology behind equiplets specifically and the infrastructure in general that is required to develop such a flexible and reconfigurable manufacturing system.

To enable Grid Manufacturing, a whole set of technological challenges has been investigated. The background, research approach and concepts cover the first three introductory chapters, where after the main research starts with Chapter 4 Object Awareness. This chapter introduces an approach to dynamically handle and localise products by combining knowledge from different autonomous systems. The chapter is followed by Chapter 5 Reconfiguration, which shows how products can communicate and control equiplets without being aware of the equiplet hardware configuration. The same chapter also shows how the hardware cannot just be used by a product, but can also be changed, making it possible to adapt the hardware configuration without re-programming the system. Subsequently, in Chapter 6 Architecture, the study

focuses on combining performance and flexibility in a hybrid architecture that is used to control the 'grid' with equiplets, showing a hybrid platform made up of both a Multi-Agent System and the Robot Operating System platform for hardware control. After the architecture is established, the thesis focuses on how the architecture can be used safely, and introduces the control systems and system behaviour to be able to predict the behaviour of the equiplets. Now these fundamental systems for Grid Manufacturing have been covered it is time to continue with Chapter 8 Validation and Utilisation, which looks from a grid perspective and shows how a grid can be used in new ways, with either a heterarchical control in which all systems are equal or not, when this is more efficient. The chapter also tests the grid with different simulated cases, to show that reconfiguration and the heterarchical approach can have several benefits.

The thesis results show that automated flexibility in manufacturing is possible with the use of autonomous reconfigurable manufacturing systems. While the entire concept needs to mature as a whole, several aspects can already be applicable in industry in the short term. This includes the method of 6D localisation by using the knowledge of different autonomous systems. Another example is a completely data-driven system, called the Hardware Abstraction Layer, for reconfigurable systems which makes it possible to change the hardware of a system without the need for new software reprogramming.

Dutch Summary

Elke periode kent zijn eigen revolutie en elke revolutie brengt zijn eigen organisatorische model met zich mee. We bevinden ons nu in de 4e industriële revolutie, waar het internet van dingen ons verbindt met autonome embedded systemen. Deze systemen zijn actief in de virtuele 'cyber' wereld, alsook in de echte 'fysieke' wereld om ons heen. Deze zogenoemde 'Cyber-Fysieke' Systemen volgen daarmee een modern organisatorisch model, namelijk zelfmanagement, en zijn dan ook in staat zelf proactieve acties te ondernemen.

Dit proefschrift belicht productiesystemen vanuit het Cyber-Fysieke perspectief. De productiesystemen zijn hier herconfigureerbaar, autonoom en zeer flexibel. Dit kan enkel worden bereikt door het ontwikkelen van nieuwe methodes en het toepassen van nieuwe technologieën die flexibiliteit verder bevorderen. Echter, efficiëntie is ook van belang, bijvoorbeeld door productassemblage zo flexibel te maken dat het daardoor kostenefficiënt is om de productie van diverse producten met een lage oplage, zogenaamde high-mix, low volume producten, te automatiseren. De mogelijkheid om zo flexibel te kunnen produceren moet bereikt worden door de creatie van nieuwe methoden en middelen, waarbij nieuwe technologieën worden gecombineerd; een belangrijk aspect hierbij is dat dit toepasbaar getest moet worden door gebruik van simulatoren en speciaal hiervoor ontwikkelde productiesystemen. Dit onderzoek zal beginnen met het introduceren van het concept achter de bijbehorende productiemethodologie, welke Grid Manufacturing is genoemd. Grid Manufacturing wordt uitgevoerd door autonome entiteiten (agenten) die zowel de productiesystemen zelf, als de producten representeren. Producten leven dan al in de virtuele cyber wereld voordat zij daadwerkelijk zijn gebouwd, en zijn zich bewust uit welke onderdelen zij gemaakt moeten worden. De producten communiceren en overleggen met de autonome herconfigureerbare productiesystemen, de zogenaamde equiplets. Deze equiplets leveren generieke diensten aan een grote diversiteit aan producten, die hierdoor op elk moment geproduceerd kunnen worden. Het onderzoek focust hierbij specifiek op de equiplets en de technische uitdagingen om dynamisch geautomatiseerde productie mogelijk te maken.

Om Grid Manufacturing mogelijk te maken is er een set van technologische uitdagingen onderzocht. De achtergrond, onderzoeksaanpak en concepten zijn dan ook de eerste drie inleidende hoofdstukken. Daarna begint het onderzoek met Hoofdstuk 4 Object Awareness. Dit hoofdstuk beschrijft een dynamische manier waarop informatie uit verschillende autonome systemen gecombineerd wordt om objecten te herkennen, lokaliseren en daarmee te kunnen manipuleren. Hoofdstuk 5 Herconfiguratie beschrijft hoe producten communiceren met de equiplets en welke achterliggende systemen ervoor zorgen dat, ondanks

dat het product niet bekend is met de hardware van de equiplet, deze toch in staat is acties uit te voeren. Tevens beschrijft het hoofdstuk hoe de equiplets omgaan met verschillende hardwareconfiguraties en ondanks de aanpassingen zichzelf toch kunnen besturen. De equiplet kan dan ook aangepast worden zonder dat deze opnieuw geprogrammeerd hoeft te worden. In Hoofdstuk 6 Architectuur wordt vervolgens dieper ingegaan op de bovenliggende architectuur van de equiplets. Hier worden prestaties gecombineerd met flexibiliteit, waarvoor een hybride architectuur is ontwikkeld die het grid van equiplets controleert door het gebruik van twee platformen: Multi-Agent System (MAS) en Robot Operating System (ROS). Nadat de architectuur is vastgesteld, wordt er in Hoofdstuk 7 onderzocht hoe deze veilig ingezet kan worden. Hierbij wordt een controlesysteem ingevoerd dat het systeemgedrag bepaalt, waarmee het gedrag van de equiplets transparant wordt gemaakt. Tevens zal een simulatie met input van de sensoren uit de fysieke wereld 'live' controleren of alle bewegingen veilig uitgevoerd kunnen worden. Nadat de basisfunctionaliteit van het Grid nu compleet is, wordt in Hoofdstuk 8 Validatie en Utilisatie gekeken naar hoe Grid Manufacturing gebruikt kan worden en welke nieuwe mogelijkheden deze kan opleveren. Zo wordt er besproken hoe zowel een hiërarchische als een heterarchische aanpak, waar alle systemen gelijk zijn, gebruikt kan worden. Daarnaast laat het hoofdstuk o.a. aan de hand van enkele voorbeelden en simulaties zien welke effecten herconfiguratie kan hebben, en welke voordelen deze aanpak zoal kan bieden..

Het proefschrift laat zien hoe met technische middelen geautomatiseerde flexibiliteit mogelijk wordt gemaakt. Hoewel het gehele concept nog volwassen zal moeten worden, worden er enkele aspecten getoond die op de korte termijn toepasbaar zijn in de industrie. Enkele voorbeelden hiervan zijn: (1) het combineren van gegevens uit diverse (autonome) bronnen voor 6D-lokalisatie; (2) een data-gedreven systeem, de zogeheten hardware-abstractielaag, die herconfigureerbare systemen controleert en de mogelijkheid biedt om deze productiesystemen aan te passen zonder deze te hoeven herprogrammeren; en (3) het gebruik van Cyber-Fysieke systemen om de veiligheid te verhogen.

Curriculum Vitae

Personal Data

Name: Daniël Harold Telgen

Birthday: 8 November 1981

Birthplace: Rotterdam, the Netherlands

Work Experience

Now - 2016: Head of Department HBO-ICT Computer Engineering and Business IT & Management at the HU University of Applied Sciences Utrecht

Now - 2012: Project Manager at the Centre for Technology and Innovation HU University of Applied Sciences Utrecht

2016 - 2012: Senior Lecturer Computer Engineering at the HU University of Applied Sciences Utrecht

2012 - 2010: Researcher & Lecturer Computer Engineering at the HU University of Applied Sciences Utrecht

2010 - 2007: Technical Consultant at Alten

2009 - 2007: Control Application Engineer at Priva

2007 - 2006: Trainee Researcher (Master Thesis) at Dalarna University, Sweden

2005 - 2004: Software Engineer at Satellite Services BV

2004 - 2004: Trainee Software Engineer at Satellite Services BV

2002 - 2001: Trainee Embedded Software Engineer at Opticon Sensors Europe B.V.

Education

2016 - 2011: PhD at the Intelligent Systems Group, Utrecht University, the Netherlands

2007 - 2005: Master of Science in Computer Engineering, speciality: Intelligent Systems at Dalarna University, Sweden

2004 - 1999: Bachelor of Engineering in Electrical Engineering, speciality: Computer Engineering at the HTS Haarlem, the Netherlands.

Acknowledgements

(in Dutch)

In 2009 liep ik voor het eerst de Hogeschool Utrecht binnen, om als industrial resident vanuit het bedrijfsleven mee te werken aan een onderzoeksproject van *lector Erik Puik*. Dit bleek een keerpunt in mijn leven en het begin van een groot avontuur, dat uiteindelijk zou leiden tot, onder andere, dit proefschrift. Toentertijd had ik nooit gedacht dat ik ooit voor de klas zou staan, het bedrijfsleven zou verlaten of het onderwijs een familieaangelegenheid zou worden, maar het liep anders! Nu, zo'n zeven jaar verder, heb ik vele avonturen mogen beleven als docent, projectleider en onderzoeker, ik ontmoette mijn verloofde, een hogeschooldocent didactiek, werd vader van twee kinderen en ik ben ondertussen al bezig met een nieuwe uitdaging: het leiden van twee teams die geweldig onderwijs en onderzoek uitvoeren. Dat biedt mijn passie voor techniek een nieuw perspectief, namelijk niet alleen bezig zijn met innovatie, onderzoek, en techniek zelf, maar ook nog eens veiligstellen dat dit onze gezamenlijk cultuur zal zijn door de ingenieurs van de toekomst te creëren!

Om in deze drukke fase van mijn leven te promoveren was dan ook een serieuze uitdaging, één die onmogelijk was geweest zonder alle geweldige mensen om mij heen. Daarom wil ik ook mijn dank uitspreken aan velen, maar in het bijzonder aan mijn steun en toeverlaat, mijn partner *José Overkamp*. Zij maakte zelfs hoogzwanger van onze tweede zoon ruimte om mij onder de druk van een uitgerekende datum te laten werken aan de laatste zware loodjes van dit proefschrift. Daarnaast had dit niet mogelijk geweest zonder de hulp van mijn ouders *Harold Telgen* en *Nelleke Telgen-Theunisse*. Samen met mijn schoonmoeder, *Conny de Vrijer*, hebben zij ons enorm ontlast de laatste jaren. Zonder deze hulp had ons leven er totaal anders uitgezien en had ik waarschijnlijk mijn proefschrift nooit af kunnen ronden. Echter, jullie waren er wel! En hierdoor hebben zowel José en ik niet alleen een geweldige baan, maar is dit proefschrift nu voltooid (voor zover een proefschrift ooit voltooid kan zijn), en hebben we ook nog eens twee geweldige vrolijke gezonde zoons, *Noah* en *Sam Telgen*, die gelukkig niets tekort zijn gekomen en elke dag, wat er ook gebeurt, weer een glimlach op mijn gezicht kunnen toveren.

Vanzelfsprekend is er veel meer waar ik dankbaar voor ben, met name natuurlijk de Hogeschool Utrecht, die mij het vertrouwen en de mogelijkheid gaf voor deze kans om part-time te promoveren. Mijn promotor *prof. dr. John-Jules Meyer*, die het zag zitten om mij als 'jonge (pragmatische) hond uit het bedrijfsleven' te begeleiden in mijn reis om wetenschapper te worden. En dan is er degene die me eigenlijk op dit pad heeft gebracht en de grootste

factor was om deze weg in te slaan, namelijk *Erik Puik*, de grondlegger, dan wel *godfather*, van agile manufacturing en vele andere innovaties binnen de Hogeschool Utrecht. Erik laat constant iedereen zien wat er mogelijk is en is daarmee een enorme drijvende kracht binnen de Hogeschool Utrecht. Zonder Erik was ik hier niet geweest, maar waar ik hem het meest dankbaar voor ben zijn de vele avonturen, de geweldige ideeën en dromen die we samen hebben gerealiseerd, de brainstormsessies en de uitdagingen die we zijn aangegaan, en de manier waarmee je mij (en anderen) tot over het randje duwde, maar nooit liet vallen. Dan ook dank aan *Leo van Moergestel*, altijd positief, en ik kon altijd bij hem terecht. Ik leerde Leo stiekem in 1999 al kennen vanuit zijn studieboeken en nu staan we samen als vrienden bij één van de mooiste hbo-opleidingen van Nederland! Tevens vind ik het nodig om niet alleen dank, maar meer een verontschuldiging aan mijn vrienden uit te spreken, die mij de laatste jaren (mede vanwege de promotie) veel te weinig hebben gezien, maar waaraan ik beloof komend jaar weer voor het eerst in jaren mijn verjaardag te vieren, en hopelijk weer wat vaker op vrijdagmiddag op te duiken in ons favoriete café om het weekend in te luiden. Specifiek wil ik ook *Maarten Dinkla* bedanken, die zijn tic om mijn magere taalgevoel te corrigeren nu eens voor het goede doel kon inzetten door mijn proefschrift te proofreaden.

Als laatste komen eigenlijk degenen die het meest hebben bijgedragen, want welke ideeën ik ook had, uiteindelijk was het project vele malen te groot om alleen uit te voeren. Zo hebben we machines (hardware) ontworpen, vele hardwaremodules gecreëerd, producten bedacht en een slordige 2500 source files aan code ontworpen en geschreven. Naar schatting bevat het project dan ook niet vier manjaren werk, maar een ruime veelvoud daarvan, en dat soms letterlijk met zweet en tranen tot stand is gekomen. Een aantal mensen wil ik dan ook specifiek noemen: *Joost van Duijn* en *Rik Lafeber*, twee geweldige werktuigbouwkundig ingenieurs en vrienden die me oneindig hebben geholpen met briljante ontwerpen, maar ook met vele klusjes ver onder hun uitstekende niveau. Tevens *Tommas Bakker*, één van de meest briljantste Technische Informatica studenten die ik tot nog toe heb mogen ontmoeten, en die essentieel was in een groot deel van de implementatie en code van de Hardware Abstraction Layer die is geschreven voor dit project. Ook L*aurens van den Brink* wil ik graag bedanken. Hij was een van de eerste master studenten die afstudeerden bij de Hogeschool Utrecht en heeft geholpen bij de implementatie van de simulaties in het laatste hoofdstuk (Validation & utilisation). Uiteindelijk hebben bijna 100 studenten een bijdrage geleverd aan dit project, die ik zo meteen zoveel mogelijk bij naam zal noemen. Allen, mijn dank! Dan wil ik natuurlijk ook de leescommissie bedanken, die de moeite hebben genomen mijn proefschrift te lezen en te beoordelen, en die dit in sommige gevallen naar een hoger niveau hebben getild: *prof. dr. Jaap van den Herik, dr. ir. Heico Sandee, prof. dr. Imre Horváth, prof. dr. Frances Brazier en prof. dr. S.*

Brinkkemper. Dank voor jullie feedback en inspanning.

Vanzelfsprekend waren er vele anderen die hielpen met hun ondersteuning: secretariaat, projectmedewerkers, onderzoekers, inkopers, voedselbezorgers, managers, lectoren, schoonmakers, verhuizers, catering, labbeheerders, diverse collegas, enzovoort. Enorm veel dank!

Ten slotte, het was een (zeer) uitdagende, maar geweldige tijd, maar ook één die ik nu graag afsluit op weg naar een nieuwe start in een andere woning, met gezin, avonturen, een nieuwe functie en andere uitdagingen. Zoals altijd stel ik mezelf wel de vraag, had ik het met de kennis van nu weer opnieuw gedaan? Het antwoord is ja. Echter niet op deze manier, zowel niet qua aanpak, als qua inhoud. Toch... misschien is dat precies waar het om gaat en kan ik dan ook vaststellen, dat het gelukt is, want ik heb enorm veel mogen leren!

Acknowledgement to the (Student) Project Teams

2011-2012
Zep Mouris, Kasper van Nieuwland, Lukas Vermond, Franc Pape, Glenn Meerstra, Wouter Langerak, Jules Blok, Pascal Muller, Martijn Beek, Geerten Klarenbeek, Rick Klomp.

2012-2013
Daan Veltman, Koen Braham, Dennis Koole, Dick van der Steen, Arjan Groenewegen, Patrick de Wit, Arjen van Zanten, Bernard van Setten, Arjen van Zanten, Ammar Abdulamir, and 22 bachelor students of the Computer Engineering Ubiquitous Computing specialisation.

2013-2014
Alexander Streng, Roy Scheefhals, Duncan Jenkins, Alexander Hustinx, Garik Hakopian, Tommas Bakker, Laurens van den Brink, Bas Voskuijlen, Tom Oosterwijk, Ari Ayala Mendoza, Martin Broers, Rolf Smit

2014-2015
Tommas Bakker, Laurens van den Brink, Lars Veenendaal, Peter Markotić, Niek Arends, Harmen Klink, Pim te Slaa, Kevin Bosman, Mitchell van Rijkom, Hendrik Cornelisse, Auke de Witte, Thomas Kok, Benno Zeeman, Jeroen Huisen.

2015-2016
Edwin Koek, Casper Wolf, Kjeld Perquin, Arnout Reitsma, Floris Rijker, Jos Roijakkers, Yorrick Lans, Bianca Krieger, Tom Verloop, Feiko Wielsma, Wibren Wiersma.

Bedankt!!!

"I come and I go, but I never Leave" *Daniël Harold Telgen*

SIKS Dissertation Series

====
2009
====

2009-01 Rasa Jurgelenaite (RUN), *Symmetric Causal Independence Models*
2009-02 Willem Robert van Hage (VU), *Evaluating Ontology-Alignment Techniques*
2009-03 Hans Stol (UvT), *A Framework for Evidence-based Policy Making Using IT*
2009-04 Josephine Nabukenya (RUN), *Improving the Quality of Organisational Policy Making using Collaboration Engineering*
2009-05 Sietse Overbeek (RUN), *Bridging Supply and Demand for Knowledge Intensive Tasks - Based on Knowledge, Cognition, and Quality*
2009-06 Muhammad Subianto (UU), *Understanding Classification*
2009-07 Ronald Poppe (UT), *Discriminative Vision-Based Recovery and Recognition of Human Motion*
2009-08 Volker Nannen (VU), *Evolutionary Agent-Based Policy Analysis in Dynamic Environments*
2009-09 Benjamin Kanagwa (RUN), *Design, Discovery and Construction of Service-oriented Systems*
2009-10 Jan Wielemaker (UVA), *Logic programming for knowledge-intensive interactive applications*
2009-11 Alexander Boer (UVA), *Legal Theory, Sources of Law & the Semantic Web*
2009-12 Peter Massuthe (TUE, Humboldt-Universitaet zu Berlin), *Operating Guidelines for Services*
2009-13 Steven de Jong (UM), *Fairness in Multi-Agent Systems*
2009-14 Maksym Korotkiy (VU), *From ontology-enabled services to service-enabled ontologies (making ontologies work in e-science with ONTO-SOA)*
2009-15 Rinke Hoekstra (UVA), *Ontology Representation - Design Patterns and Ontologies that Make Sense*
2009-16 Fritz Reul (UvT), *New Architectures in Computer Chess*
2009-17 Laurens van der Maaten (UvT), *Feature Extraction from Visual Data*
2009-18 Fabian Groffen (CWI), *Armada, An Evolving Database System*
2009-19 Valentin Robu (CWI), *Modeling Preferences, Strategic Reasoning and Collaboration in Agent-Mediated Electronic Markets*
2009-20 Bob van der Vecht (UU), *Adjustable Autonomy: Controling Influences on Decision Making*
2009-21 Stijn Vanderlooy (UM), *Ranking and Reliable Classification*
2009-22 Pavel Serdyukov (UT), *Search For Expertise: Going beyond direct evidence*
2009-23 Peter Hofgesang (VU), *Modelling Web Usage in a Changing Environment*
2009-24 Annerieke Heuvelink (VUA), *Cognitive Models for Training Simulations*
2009-25 Alex van Ballegooij (CWI), *"RAM: Array Database Management through Relational Mapping"*
2009-26 Fernando Koch (UU), *An Agent-Based Model for the Development of Intelligent Mobile Services*
2009-27 Christian Glahn (OU), *Contextual Support of social Engagement and Reflection on the Web*
2009-28 Sander Evers (UT), *Sensor Data Management with Probabilistic Models*
2009-29 Stanislav Pokraev (UT), *Model-Driven Semantic Integration of Service-Oriented Applications*
2009-30 Marcin Zukowski (CWI), *Balancing vectorized query execution with bandwidth-optimized storage*
2009-31 Sofiya Katrenko (UVA), *A Closer Look at Learning Relations from Text*
2009-32 Rik Farenhorst (VU) and Remco de Boer (VU), *Architectural Knowledge Management: Supporting Architects and Auditors*
2009-33 Khiet Truong (UT), *How Does Real Affect Affect Affect Recognition In Speech?*
2009-34 Inge van de Weerd (UU), *Advancing in Software Product Management: An Incremental Method Engineering Approach*
2009-35 Wouter Koelewijn (UL), *Privacy en Politiegegevens; Over geautomatiseerde normatieve informatie-uitwisseling*
2009-36 Marco Kalz (OUN), *Placement Support for Learners in Learning Networks*
2009-37 Hendrik Drachsler (OUN), *Navigation Support for Learners in Informal Learning Networks*
2009-38 Riina Vuorikari (OU), *Tags and self-organisation: a metadata ecology for learning resources in a multilingual context*
2009-39 Christian Stahl (TUE, Humboldt-Universitaet zu Berlin), *Service Substitution – A Behavioral Approach Based on Petri Nets*
2009-40 Stephan Raaijmakers (UvT), *Multinomial Language Learning: Investigations into the Geometry of Language*
2009-41 Igor Berezhnyy (UvT), *Digital Analysis of Paintings*
2009-42 Toine Bogers (UvT), *Recommender Systems for Social Bookmarking*
2009-43 Virginia Nunes Leal Franqueira (UT), *Finding Multi-step Attacks in Computer Networks using Heuristic Search and Mobile Ambients*
2009-44 Roberto Santana Tapia (UT), *Assessing Business-IT Alignment in Networked Organizations*
2009-45 Jilles Vreeken (UU), *Making Pattern Mining Useful*
2009-46 Loredana Afanasiev (UvA), *Querying XML: Benchmarks and Recursion*

====
2010
====

2010-01 Matthijs van Leeuwen (UU), *Patterns that Matter*
2010-02 Ingo Wassink (UT), *Work flows in Life Science*
2010-03 Joost Geurts (CWI), *A Document Engineering Model and Processing Framework for Multimedia documents*
2010-04 Olga Kulyk (UT), *Do You Know What I Know? Situational Awareness of Co-located Teams in Multidisplay Environments*
2010-05 Claudia Hauff (UT), *Predicting the Effectiveness of Queries and Retrieval Systems*
2010-06 Sander Bakkes (UvT), *Rapid Adaptation of Video Game AI*
2010-07 Wim Fikkert (UT), *Gesture interaction at a Distance*
2010-08 Krzysztof Siewicz (UL), *Towards an Improved Regulatory Framework of Free Software. Protecting user freedoms in a world of software communities and eGovernments*
2010-09 Hugo Kielman (UL), *Politiele gegevensverwerking en Privacy, Naar een effectieve waarborging*

2010-10 Rebecca Ong (UL), *Mobile Communication and Protection of Children*
2010-11 Adriaan Ter Mors (TUD), *The world according to MARP: Multi-Agent Route Planning*
2010-12 Susan van den Braak (UU), *Sensemaking software for crime analysis*
2010-13 Gianluigi Folino (RUN), *High Performance Data Mining using Bio-inspired techniques*
2010-14 Sander van Splunter (VU), *Automated Web Service Reconfiguration*
2010-15 Lianne Bodenstaff (UT), *Managing Dependency Relations in Inter-Organizational Models*
2010-16 Sicco Verwer (TUD), *Efficient Identification of Timed Automata, theory and practice*
2010-17 Spyros Kotoulas (VU), *Scalable Discovery of Networked Resources: Algorithms, Infrastructure, Applications*
2010-18 Charlotte Gerritsen (VU), *Caught in the Act: Investigating Crime by Agent-Based Simulation*
2010-19 Henriette Cramer (UvA), *People's Responses to Autonomous and Adaptive Systems*
2010-20 Ivo Swartjes (UT), *Whose Story Is It Anyway? How Improv Informs Agency and Authorship of Emergent Narrative*
2010-21 Harold van Heerde (UT), *Privacy-aware data management by means of data degradation*
2010-22 Michiel Hildebrand (CWI), *End-user Support for Access to Heterogeneous Linked Data*
2010-23 Bas Steunebrink (UU), *The Logical Structure of Emotions*
2010-24 Dmytro Tykhonov, *Designing Generic and Efficient Negotiation Strategies*
2010-25 Zulfiqar Ali Memon (VU), *Modelling Human-Awareness for Ambient Agents: A Human Mindreading Perspective*
2010-26 Ying Zhang (CWI), *XRPC: Efficient Distributed Query Processing on Heterogeneous XQuery Engines*
2010-27 Marten Voulon (UL), *Automatisch contracteren*
2010-28 Arne Koopman (UU), *Characteristic Relational Patterns*
2010-29 Stratos Idreos(CWI), *Database Cracking: Towards Auto-tuning Database Kernels*
2010-30 Marieke van Erp (UvT), *Accessing Natural History - Discoveries in data cleaning, structuring, and retrieval*
2010-31 Victor de Boer (UVA), *Ontology Enrichment from Heterogeneous Sources on the Web*
2010-32 Marcel Hiel (UvT), *An Adaptive Service Oriented Architecture: Automatically solving Interoperability Problems*
2010-33 Robin Aly (UT), *Modeling Representation Uncertainty in Concept-Based Multimedia Retrieval*
2010-34 Teduh Dirgahayu (UT), *Interaction Design in Service Compositions*
2010-35 Dolf Trieschnigg (UT), *Proof of Concept: Concept-based Biomedical Information Retrieval*
2010-36 Jose Janssen (OU), *Paving the Way for Lifelong Learning; Facilitating competence development through a learning path specification*
2010-37 Niels Lohmann (TUE), *Correctness of services and their composition*
2010-38 Dirk Fahland (TUE), *From Scenarios to components*
2010-39 Ghazanfar Farooq Siddiqui (VU), *Integrative modeling of emotions in virtual agents*
2010-40 Mark van Assem (VU), *Converting and Integrating Vocabularies for the Semantic Web*
2010-41 Guillaume Chaslot (UM), *Monte-Carlo Tree Search*
2010-42 Sybren de Kinderen (VU), *Needs-driven service bundling in a multi-supplier setting - the computational e3-service approach*
2010-43 Peter van Kranenburg (UU), *A Computational Approach to Content-Based Retrieval of Folk Song Melodies*
2010-44 Pieter Bellekens (TUE), *An Approach towards Context-sensitive and User-adapted Access to Heterogeneous Data Sources, Illustrated in the Television Domain*
2010-45 Vasilios Andrikopoulos (UvT), *A theory and model for the evolution of software services*
2010-46 Vincent Pijpers (VU), *e3alignment: Exploring Inter-Organizational Business-ICT Alignment*
2010-47 Chen Li (UT), *Mining Process Model Variants: Challenges, Techniques, Examples*
2010-48 Withdrawn
2010-49 Jahn-Takeshi Saito (UM), *Solving difficult game positions*
2010-50 Bouke Huurnink (UVA), *Search in Audiovisual Broadcast Archives*
2010-51 Alia Khairia Amin (CWI), *Understanding and supporting information seeking tasks in multiple sources*
2010-52 Peter-Paul van Maanen (VU), *Adaptive Support for Human-Computer Teams: Exploring the Use of Cognitive Models of Trust and Attention*
2010-53 Edgar Meij (UVA), *Combining Concepts and Language Models for Information Access*

====
2011
====

2011-01 Botond Cseke (RUN), *Variational Algorithms for Bayesian Inference in Latent Gaussian Models*
2011-02 Nick Tinnemeier(UU), *Organizing Agent Organizations. Syntax and Operational Semantics of an Organization-Oriented Programming Language*
2011-03 Jan Martijn van der Werf (TUE), *Compositional Design and Verification of Component-Based Information Systems*
2011-04 Hado van Hasselt (UU), *Insights in Reinforcement Learning; Formal analysis and empirical evaluation of temporal-difference learning algorithms*
2011-05 Base van der Raadt (VU), *Enterprise Architecture Coming of Age - Increasing the Performance of an Emerging Discipline.*
2011-06 Yiwen Wang (TUE), *Semantically-Enhanced Recommendations in Cultural Heritage*
2011-07 Yujia Cao (UT), *Multimodal Information Presentation for High Load Human Computer Interaction*
2011-08 Nieske Vergunst (UU), *BDI-based Generation of Robust Task-Oriented Dialogues*
2011-09 Tim de Jong (OU), *Contextualised Mobile Media for Learning*
2011-10 Bart Bogaert (UvT), *Cloud Content Contention*
2011-11 Dhaval Vyas (UT), *Designing for Awareness: An Experience-focused HCI Perspective*
2011-12 Carmen Bratosin (TUE), *Grid Architecture for Distributed Process Mining*
2011-13 Xiaoyu Mao (UvT), *Airport under Control. Multiagent Scheduling for Airport Ground Handling*
2011-14 Milan Lovric (EUR), *Behavioral Finance and Agent-Based Artificial Markets*
2011-15 Marijn Koolen (UvA), *The Meaning of Structure: the Value of Link Evidence for Information Retrieval*
2011-16 Maarten Schadd (UM), *Selective Search in Games of Different Complexity*
2011-17 Jiyin He (UVA), *Exploring Topic Structure: Coherence, Diversity and Relatedness*
2011-18 Mark Ponsen (UM), *Strategic Decision-Making in complex games*
2011-19 Ellen Rusman (OU), *The Mind's Eye on Personal Profiles*
2011-20 Qing Gu (VU), *Guiding service-oriented software engineering - A view-based approach*

2011-21 Linda Terlouw (TUD), *Modularization and Specification of Service-Oriented Systems*
2011-22 Junte Zhang (UVA), *System Evaluation of Archival Description and Access*
2011-23 Wouter Weerkamp (UVA), *Finding People and their Utterances in Social Media*
2011-24 Herwin van Welbergen (UT), *Behavior Generation for Interpersonal Coordination with Virtual Humans On Specifying, Scheduling and Realizing Multimodal Virtual Human Behavior*
2011-25 Syed Waqar ul Qounain Jaffry (VU), *Analysis and Validation of Models for Trust Dynamics*
2011-26 Matthijs Aart Pontier (VU), *Virtual Agents for Human Communication - Emotion Regulation and Involvement-Distance Trade-Offs in Embodied Conversational Agents and Robots*
2011-27 Aniel Bhulai (VU), *Dynamic website optimization through autonomous management of design patterns*
2011-28 Rianne Kaptein(UVA), *Effective Focused Retrieval by Exploiting Query Context and Document Structure*
2011-29 Faisal Kamiran (TUE), *Discrimination-aware Classification*
2011-30 Egon van den Broek (UT), *Affective Signal Processing (ASP): Unraveling the mystery of emotions*
2011-31 Ludo Waltman (EUR), *Computational and Game-Theoretic Approaches for Modeling Bounded Rationality*
2011-32 Nees-Jan van Eck (EUR), *Methodological Advances in Bibliometric Mapping of Science*
2011-33 Tom van der Weide (UU), *Arguing to Motivate Decisions*
2011-34 Paolo Turrini (UU), *Strategic Reasoning in Interdependence: Logical and Game-theoretical Investigations*
2011-35 Maaike Harbers (UU), *Explaining Agent Behavior in Virtual Training*
2011-36 Erik van der Spek (UU), *Experiments in serious game design: a cognitive approach*
2011-37 Adriana Burlutiu (RUN), *Machine Learning for Pairwise Data, Applications for Preference Learning and Supervised Network Inference*
2011-38 Nyree Lemmens (UM), *Bee-inspired Distributed Optimization*
2011-39 Joost Westra (UU), *Organizing Adaptation using Agents in Serious Games*
2011-40 Viktor Clerc (VU), *Architectural Knowledge Management in Global Software Development*
2011-41 Luan Ibraimi (UT), *Cryptographically Enforced Distributed Data Access Control*
2011-42 Michal Sindlar (UU), *Explaining Behavior through Mental State Attribution*
2011-43 Henk van der Schuur (UU), *Process Improvement through Software Operation Knowledge*
2011-44 Boris Reuderink (UT), Robust *Brain-Computer Interfaces*
2011-45 Herman Stehouwer (UvT), *Statistical Language Models for Alternative Sequence Selection*
2011-46 Beibei Hu (TUD), *Towards Contextualized Information Delivery: A Rule-based Architecture for the Domain of Mobile Police Work*
2011-47 Azizi Bin Ab Aziz (VU), *Exploring Computational Models for Intelligent Support of Persons with Depression*
2011-48 Mark Ter Maat (UT), *Response Selection and Turn-taking for a Sensitive Artificial Listening Agent*
2011-49 Andreea Niculescu (UT), *Conversational interfaces for task-oriented spoken dialogues: design aspects influencing interaction quality*

====
2012
====

2012-01 Terry Kakeeto (UvT), *Relationship Marketing for SMEs in Uganda*
2012-02 Muhammad Umair(VU), *Adaptivity, emotion, and Rationality in Human and Ambient Agent Models*
2012-03 Adam Vanya (VU), *Supporting Architecture Evolution by Mining Software Repositories*
2012-04 Jurriaan Souer (UU), *Development of Content Management System-based Web Applications*
2012-05 Marijn Plomp (UU), *Maturing Interorganisational Information Systems*
2012-06 Wolfgang Reinhardt (OU), *Awareness Support for Knowledge Workers in Research Networks*
2012-07 Rianne van Lambalgen (VU), *When the Going Gets Tough: Exploring Agent-based Models of Human Performance under Demanding Conditions*
2012-08 Gerben de Vries (UVA), *Kernel Methods for Vessel Trajectories*
2012-09 Ricardo Neisse (UT), *Trust and Privacy Management Support for Context-Aware Service Platforms*
2012-10 David Smits (TUE), *Towards a Generic Distributed Adaptive Hypermedia Environment*
2012-11 J.C.B. Rantham Prabhakara (TUE), *Process Mining in the Large: Preprocessing, Discovery, and Diagnostics*
2012-12 Kees van der Sluijs (TUE), *Model Driven Design and Data Integration in Semantic Web Information Systems*
2012-13 Suleman Shahid (UvT), F*un and Face: Exploring non-verbal expressions of emotion during playful interactions*
2012-14 Evgeny Knutov(TUE), *Generic Adaptation Framework for Unifying Adaptive Web-based Systems*
2012-15 Natalie van der Wal (VU), *Social Agents. Agent-Based Modelling of Integrated Internal and Social Dynamics of Cognitive and Affective Processes.*
2012-16 Fiemke Both (VU), *Helping people by understanding them - Ambient Agents supporting task execution and depression treatment*
2012-17 Amal Elgammal (UvT), *Towards a Comprehensive Framework for Business Process Compliance*
2012-18 Eltjo Poort (VU), *Improving Solution Architecting Practices*
2012-19 Helen Schonenberg (TUE), *What's Next? Operational Support for Business Process Execution*
2012-20 Ali Bahramisharif (RUN), *Covert Visual Spatial Attention, a Robust Paradigm for Brain-Computer Interfacing*
2012-21 Roberto Cornacchia (TUD), *Querying Sparse Matrices for Information Retrieval*
2012-22 Thijs Vis (UvT), *Intelligence, politie en veiligheidsdienst: verenigbare grootheden?*
2012-23 Christian Muehl (UT), *Toward Affective Brain-Computer Interfaces: Exploring the Neurophysiology of Affect during Human Media Interaction*
2012-24 Laurens van der Werff (UT), *Evaluation of Noisy Transcripts for Spoken Document Retrieval*
2012-25 Silja Eckartz (UT), *Managing the Business Case Development in Inter-Organizational IT Projects: A Methodology and its Application*
2012-26 Emile de Maat (UVA), *Making Sense of Legal Text*
2012-27 Hayrettin Gurkok (UT), *Mind the Sheep! User Experience Evaluation & Brain-Computer Interface Games*
2012-28 Nancy Pascall (UvT), *Engendering Technology Empowering Women*
2012-29 Almer Tigelaar (UT), *Peer-to-Peer Information Retrieval*
2012-30 Alina Pommeranz (TUD), *Designing Human-Centered Systems for Reflective Decision Making*
2012-31 Emily Bagarukayo (RUN), *A Learning by Construction Approach for Higher Order Cognitive Skills Improvement, Building Capacity and Infrastructure*
2012-32 Wietske Visser (TUD), *Qualitative multi-criteria preference representation and reasoning*
2012-33 Rory Sie (OUN), *Coalitions in Cooperation Networks (COCOON)*
2012-34 Pavol Jancura (RUN), *Evolutionary analysis in PPI networks and applications*

S

2012-35 Evert Haasdijk (VU), *Never Too Old To Learn – On-line Evolution of Controllers in Swarm- and Modular Robotics*
2012-36 Denis Ssebugwawo (RUN), *Analysis and Evaluation of Collaborative Modeling Processes*
2012-37 Agnes Nakakawa (RUN), *A Collaboration Process for Enterprise Architecture Creation*
2012-38 Selmar Smit (VU), *Parameter Tuning and Scientific Testing in Evolutionary Algorithms*
2012-39 Hassan Fatemi (UT), *Risk-aware design of value and coordination networks*
2012-40 Agus Gunawan (UvT), *Information Access for SMEs in Indonesia*
2012-41 Sebastian Kelle (OU), *Game Design Patterns for Learning*
2012-42 Dominique Verpoorten (OU), *Reflection Amplifiers in self-regulated Learning*
2012-43 Withdrawn
2012-44 Anna Tordai (VU), *On Combining Alignment Techniques*
2012-45 Benedikt Kratz (UvT), *A Model and Language for Business-aware Transactions*
2012-46 Simon Carter (UVA), *Exploration and Exploitation of Multilingual Data for Statistical Machine Translation*
2012-47 Manos Tsagkias (UVA), *Mining Social Media: Tracking Content and Predicting Behavior*
2012-48 Jorn Bakker (TUE), *Handling Abrupt Changes in Evolving Time-series Data*
2012-49 Michael Kaisers (UM), *Learning against Learning - Evolutionary dynamics of reinforcement learning algorithms in strategic interactions*
2012-50 Steven van Kervel (TUD), *Ontologogy driven Enterprise Information Systems Engineering*
2012-51 Jeroen de Jong (TUD), *Heuristics in Dynamic Sceduling; a practical framework with a case study in elevator dispatching*

====
2013
====

2013-01 Viorel Milea (EUR), *News Analytics for Financial Decision Support*
2013-02 Erietta Liarou (CWI), *MonetDB/DataCell: Leveraging the Column-store Database Technology for Efficient and Scalable Stream Processing*
2013-03 Szymon Klarman (VU), *Reasoning with Contexts in Description Logics*
2013-04 Chetan Yadati (TUD), *Coordinating autonomous planning and scheduling*
2013-05 Dulce Pumareja (UT), *Groupware Requirements Evolutions Patterns*
2013-06 Romulo Goncalves (CWI), *The Data Cyclotron: Juggling Data and Queries for a Data Warehouse Audience*
2013-07 Giel van Lankveld (UvT), *Quantifying Individual Player Differences*
2013-08 Robbert-Jan Merk (VU), *Making enemies: cognitive modeling for opponent agents in fighter pilot simulators*
2013-09 Fabio Gori (RUN), *Metagenomic Data Analysis: Computational Methods and Applications*
2013-10 *Jeewanie Jayasinghe Arachchige (UvT), A Unified Modeling Framework for Service Design.*
2013-11 Evangelos Pournaras (TUD), *Multi-level Reconfigurable Self-organization in Overlay Services*
2013-12 Marian Razavian (VU), *Knowledge-driven Migration to Services*
2013-13 Mohammad Safiri (UT), *Service Tailoring: User-centric creation of integrated IT-based homecare services to support independent living of elderly*
2013-14 Jafar Tanha (UVA), *Ensemble Approaches to Semi-Supervised Learning Learning*
2013-15 Daniel Hennes (UM), *Multiagent Learning - Dynamic Games and Applications*
2013-16 Eric Kok (UU), *Exploring the practical benefits of argumentation in multi-agent deliberation*
2013-17 Koen Kok (VU), *The PowerMatcher: Smart Coordination for the Smart Electricity Grid*
2013-18 Jeroen Janssens (UvT), *Outlier Selection and One-Class Classification*
2013-19 Renze Steenhuizen (TUD), *Coordinated Multi-Agent Planning and Scheduling*
2013-20 Katja Hofmann (UvA), *Fast and Reliable Online Learning to Rank for Information Retrieval*
2013-21 Sander Wubben (UvT), *Text-to-text generation by monolingual machine translation*
2013-22 Tom Claassen (RUN), *Causal Discovery and Logic*
2013-23 Patricio de Alencar Silva (UvT), *Value Activity Monitoring*
2013-24 Haitham Bou Ammar (UM), *Automated Transfer in Reinforcement Learning*
2013-25 Agnieszka Anna Latoszek-Berendsen (UM), *Intention-based Decision Support. A new way of representing and implementing clinical guidelines in a Decision Support System*
2013-26 Alireza Zarghami (UT), *Architectural Support for Dynamic Homecare Service Provisioning*
2013-27 Mohammad Huq (UT), *Inference-based Framework Managing Data Provenance*
2013-28 Frans van der Sluis (UT), *When Complexity becomes Interesting: An Inquiry into the Information eXperience*
2013-29 Iwan de Kok (UT), *Listening Heads*
2013-30 Joyce Nakatumba (TUE), *Resource-Aware Business Process Management: Analysis and Support*
2013-31 Dinh Khoa Nguyen (UvT), *Blueprint Model and Language for Engineering Cloud Applications*
2013-32 Kamakshi Rajagopal (OUN), *Networking For Learning; The role of Networking in a Lifelong Learner's Professional Development*
2013-33 Qi Gao (TUD), *User Modeling and Personalization in the Microblogging Sphere*
2013-34 Kien Tjin-Kam-Jet (UT), *Distributed Deep Web Search*
2013-35 Abdallah El Ali (UvA), *Minimal Mobile Human Computer Interaction*
2013-36 Than Lam Hoang (TUe), *Pattern Mining in Data Streams*
2013-37 Dirk Brner (OUN), *Ambient Learning Displays*
2013-38 Eelco den Heijer (VU), *Autonomous Evolutionary Art*
2013-39 Joop de Jong (TUD), *A Method for Enterprise Ontology based Design of Enterprise Information Systems*
2013-40 Pim Nijssen (UM), *Monte-Carlo Tree Search for Multi-Player Games*
2013-41 Jochem Liem (UVA), *Supporting the Conceptual Modelling of Dynamic Systems: A Knowledge Engineering Perspective on Qualitative Reasoning*
2013-42 Lon Planken (TUD), *Algorithms for Simple Temporal Reasoning*
2013-43 Marc Bron (UVA), *Exploration and Contextualization through Interaction and Concepts*

====
2014
====

2014-01 Nicola Barile (UU), *Studies in Learning Monotone Models from Data*
2014-02 Fiona Tuliyano (RUN), *Combining System Dynamics with a Domain Modeling Method*

2014-03 Sergio Raul Duarte Torres (UT), *Information Retrieval for Children: Search Behavior and Solutions*
2014-04 Hanna Jochmann-Mannak (UT), *Websites for children: search strategies and interface design - Three studies on children's search performance and evaluation*
2014-05 Jurriaan van Reijsen (UU), *Knowledge Perspectives on Advancing Dynamic Capability*
2014-06 Damian Tamburri (VU), *Supporting Networked Software Development*
2014-07 Arya Adriansyah (TUE), *Aligning Observed and Modeled Behavior*
2014-08 Samur Araujo (TUD), *Data Integration over Distributed and Heterogeneous Data Endpoints*
2014-09 Philip Jackson (UvT), *Toward Human-Level Artificial Intelligence: Representation and Computation of Meaning in Natural Language*
2014-10 Ivan Salvador Razo Zapata (VU), Service Value Networks
2014-11 Janneke van der Zwaan (TUD), *An Empathic Virtual Buddy for Social Support*
2014-12 Willem van Willigen (VU), *Look Ma, No Hands: Aspects of Autonomous Vehicle Control*
2014-13 Arlette van Wissen (VU), *Agent-Based Support for Behavior Change: Models and Applications in Health and Safety Domains*
2014-14 Yangyang Shi (TUD), *Language Models With Meta-information*
2014-15 Natalya Mogles (VU), *Agent-Based Analysis and Support of Human Functioning in Complex Socio-Technical Systems: Applications in Safety and Healthcare*
2014-16 Krystyna Milian (VU), *Supporting trial recruitment and design by automatically interpreting eligibility criteria*
2014-17 Kathrin Dentler (VU), *Computing healthcare quality indicators automatically: Secondary Use of Patient Data and Semantic Interoperability*
2014-18 Mattijs Ghijsen (VU), *Methods and Models for the Design and Study of Dynamic Agent Organizations*
2014-19 Vincius Ramos (TUE), *Adaptive Hypermedia Courses: Qualitative and Quantitative Evaluation and Tool Support*
2014-20 Mena Habib (UT), *Named Entity Extraction and Disambiguation for Informal Text: The Missing Link*
2014-21 Kassidy Clark (TUD), *Negotiation and Monitoring in Open Environments*
2014-22 Marieke Peeters (UT), *Personalized Educational Games - Developing agent-supported scenario-based training*
2014-23 Eleftherios Sidirourgos (UvA/CWI), *Space Efficient Indexes for the Big Data Era*
2014-24 Davide Ceolin (VU), *Trusting Semi-structured Web Data*
2014-25 Martijn Lappenschaar (RUN), *New network models for the analysis of disease interaction*
2014-26 Tim Baarslag (TUD), *What to Bid and When to Stop*
2014-27 Rui Jorge Almeida (EUR), *Conditional Density Models Integrating Fuzzy and Probabilistic Representations of Uncertainty*
2014-28 Anna Chmielowiec (VU), *Decentralized k-Clique Matching*
2014-29 Jaap Kabbedijk (UU), *Variability in Multi-Tenant Enterprise Software*
2014-30 Peter de Cock (UvT), *Anticipating Criminal Behaviour*
2014-31 Leo van Moergestel (UU), *Agent Technology in Agile Multiparallel Manufacturing and Product Support*
2014-32 Naser Ayat (UvA), *On Entity Resolution in Probabilistic Data*
2014-33 Tesfa Tegegne (RUN), *Service Discovery in eHealth*
2014-34 Christina Manteli(VU), *The Effect of Governance in Global Software Development: Analyzing Transactive Memory Systems.*
2014-35 Joost van Ooijen (UU), *Cognitive Agents in Virtual Worlds: A Middleware Design Approach*
2014-36 Joos Buijs (TUE), *Flexible Evolutionary Algorithms for Mining Structured Process Models*
2014-37 Maral Dadvar (UT), *Experts and Machines United Against Cyberbullying*
2014-38 Danny Plass-Oude Bos (UT), *Making brain-computer interfaces better: improving usability through post-processing.*
2014-39 Jasmina Maric (UvT), *Web Communities, Immigration, and Social Capital*
2014-40 Walter Omona (RUN), *A Framework for Knowledge Management Using ICT in Higher Education*
2014-41 Frederic Hogenboom (EUR), *Automated Detection of Financial Events in News Text*
2014-42 Carsten Eijckhof (CWI/TUD), *Contextual Multidimensional Relevance Models*
2014-43 Kevin Vlaanderen (UU), *Supporting Process Improvement using Method Increments*
2014-44 Paulien Meesters (UvT), *Intelligent Blauw. Met als ondertitel: Intelligence-gestuurde politiezorg in gebiedsgebonden eenheden.*
2014-45 Birgit Schmitz (OUN), *Mobile Games for Learning: A Pattern-Based Approach*
2014-46 Ke Tao (TUD), So*cial Web Data Analytics: Relevance, Redundancy, Diversity*
2014-47 Shangsong Liang (UVA), *Fusion and Diversification in Information Retrieval*

====
2015
====

2015-01 Niels Netten (UvA), *Machine Learning for Relevance of Information in Crisis Response*
2015-02 Faiza Bukhsh (UvT), *Smart auditing: Innovative Compliance Checking in Customs Controls*
2015-03 Twan van Laarhoven (RUN), *Machine learning for network data*
2015-04 Howard Spoelstra (OUN), *Collaborations in Open Learning Environments*
2015-05 Christoph Bösch(UT), *Cryptographically Enforced Search Pattern Hiding*
2015-06 Farideh Heidari (TUD), *Business Process Quality Computation - Computing Non-Functional Requirements to Improve Business Processes*
2015-07 Maria-Hendrike Peetz(UvA), *Time-Aware Online Reputation Analysis*
2015-08 Jie Jiang (TUD), *Organizational Compliance: An agent-based model for designing and evaluating organizational interactions*
2015-09 Randy Klaassen(UT), *HCI Perspectives on Behavior Change Support Systems*
2015-10 Henry Hermans (OUN), *OpenU: design of an integrated system to support lifelong learning*
2015-11 Yongming Luo(TUE), *Designing algorithms for big graph datasets: A study of computing bisimulation and joins*
2015-12 Julie M. Birkholz (VU), *Modi Operandi of Social Network Dynamics: The Effect of Context on Scientific Collaboration Networks*
2015-13 Giuseppe Procaccianti(VU), *Energy-Efficient Software*
2015-14 Bart van Straalen (UT), *A cognitive approach to modeling bad news conversations*
2015-15 Klaas Andries de Graaf (VU), *Ontology-based Software Architecture Documentation*
2015-16 Changyun Wei (UT), *Cognitive Coordination for Cooperative Multi-Robot Teamwork*

2015-17 Andre van Cleeff (UT), *Physical and Digital Security Mechanisms: Properties, Combinations and Trade-offs*
2015-18 Holger Pirk (CWI), *Waste Not, Want Not! - Managing Relational Data in Asymmetric Memories*
2015-19 Bernardo Tabuenca (OUN), *Ubiquitous Technology for Lifelong Learners*
2015-20 Loïs Vanhee (UU), *Using Culture and Values to Support Flexible Coordination*
2015-21 Sibren Fetter (OUN), *Using Peer-Support to Expand and Stabilize Online Learning*
2015-22 Zhemin Zhu (UT), *Co-occurrence Rate Networks*
2015-23 Luit Gazendam (VU), *Cataloguer Support in Cultural Heritage*
2015-24 Richard Berendsen (UVA), *Finding People, Papers, and Posts: Vertical Search Algorithms and Evaluation*
2015-25 Steven Woudenberg (UU), *Bayesian Tools for Early Disease Detection*
2015-26 Alexander Hogenboom (EUR), *Sentiment Analysis of Text Guided by Semantics and Structure*
2015-27 Sandor Heman (CWI), *Updating compressed colomn stores*
2015-28 Janet Bagorogoza (TiU), *Knowledge management and high performance; The Uganda Financial Institutions Model for HPO*
2015-29 Hendrik Baier (UM), *Monte-Carlo Tree Search Enhancements for One-Player and Two-Player Domains*
2015-30 Kiavash Bahreini (OU), *Real-time Multimodal Emotion Recognition in E-Learning*
2015-31 Yakup Koc (TUD), *On the robustness of Power Grids*
2015-32 Jerome Gard (UL), *Corporate Venture Management in SMEs*
2015-33 Frederik Schadd (TUD), *Ontology Mapping with Auxiliary Resources*
2015-34 Victor de Graaf (UT), *Gesocial Recommender Systems*
2015-35 Jungxao Xu (TUD), *Affective Body Language of Humanoid Robots: Perception and Effects in Human Robot Interaction*

====
2016
====

2016-01 Syed Saiden Abbas (RUN), *Recognition of Shapes by Humans and Machines*
2016-02 Michiel Meulendijk (UU), *Optimizing medication reviews through decision support: prescribing a better pill to swallow*
2016-03 Maya Sappelli (RUN), *Knowledge Work in Context: User Centered Knowledge Worker Support*
2016-04 Laurens Rietveld (VU), *Publishing and Consuming Linked Data*
2016-05 Evgeny Sherkhonov (UVA), *Expanded Acyclic Queries: Containment and an Application in Explaining Missing Answers*
2016-06 Michel Wilson (TUD), *Robust scheduling in an uncertain environment*
2016-07 Jeroen de Man (VU), *Measuring and modeling negative emotions for virtual training*
2016-08 Matje van de Camp (TiU), *A Link to the Past: Constructing Historical Social Networks from Unstructured Data*
2016-09 Archana Nottamkandath (VU), *Trusting Crowdsourced Information on Cultural Artefacts*
2016-10 George Karafotias (VU), *Parameter Control for Evolutionary Algorithms*
2016-11 Anne Schuth (UVA), *Search Engines that Learn from Their Users*
2016-12 Max Knobbout (UU), *Logics for Modelling and Verifying Normative Multi-Agent Systems*
2016-13 Nana Baah Gyan (VU), *The Web, Speech Technologies and Rural Development in West Africa - An ICT4D Approach*
2016-14 Ravi Khadka(UU), *Revisiting Legacy Software System Modernization*
2016-15 Steffen Michels (RUN), *Hybrid Probabilistic Logics - Theoretical Aspects, Algorithms and Experiments*
2016-16 Guangliang Li (UVA), *Socially Intelligent Autonomous Agents that Learn from Human Reward*
2016-17 Berend Weel (VU), *Towards Embodied Evolution of Robot Organisms*
2016-18 Albert Meroño Penuela (VU), *Refining Statistical Data on the Web*
2016-19 Berend Weel (VU), *Towards Embodied Evolution of Robot Organisms*
2016-20 Albert Meroo Peuela (VU), *Refining Statistical Data on the Web*
2016-21 Alejandro Moreno Clleri (UT), *From Traditional to Interactive Playspaces: Automatic Analysis of Player Behavior in the Interactive Tag Playground*
2016-22 Grace Lewis (VU), *Software Architecture Strategies for Cyber-Foraging Systems*
2016-23 Fei Cai (UVA), *Query Auto Completion in Information Retrieval*
2016-24 Brend Wanders (UT), *Repurposing and Probabilistic Integration of Data; An Iterative and data model independent approach*
2016-25 Julia Kiseleva (TU/e), *Using Contextual Information to Understand Searching and Browsing Behavior*
2016-26 Dilhan Thilakarathne (VU), *In or Out of Control: Exploring Computational Models to Study the Role of Human Awareness and Control in Behavioural Choices, with Applications in Aviation and Energy Management Domains*
2016-27 Wen Li (TUD), *Understanding Geo-spatial Information on Social Media*
2016-28 Mingxin Zhang (TUD) *Large-scale Agent-based Social Simulation - A study on epidemic prediction and control*
2016-29 Nicolas Höning (TUD), *Peak reduction in decentralised electricity systems -Markets and prices for flexible planning*
2016-30 Ruud Mattheij (UvT), *The Eyes Have It*
2016-31 Mohammad Khelghati (UT), *Deep web content monitoring*
2016-32 Eelco Vriezekolk (UT), *Assessing Telecommunication Service Availability Risks for Crisis Organisations*
2016-33 Peter Bloem (UVA), *Single Sample Statistics, exercises in learning from just one example*
2016-34 Dennis Schunselaar (TUE), *Configurable Process Trees: Elicitation, Analysis, and Enactment*
2016-35 Zhaochun Ren (UVA), *Monitoring Social Media: Summarization, Classification and Recommendation*
2016-36 Daphne Karreman (UT), *Beyond R2D2: The design of nonverbal interaction behavior optimized for robot-specific morphologies*
2016-37 Giovanni Sileno (UvA), *Aligning Law and Action - a conceptual and computational inquiry*
2016-38 Andrea Minuto (UT), *MATERIALS THAT MATTER - Smart Materials meet Art & Interaction Design*
2016-39 Merijn Bruijnes (UT), *Believable Suspect Agents; Response and Interpersonal Style Selection for an Artificial Suspect*
2016-40 Christian Detweiler (TUD), Accounting for Values in Design
2016-41 Thomas King (TUD), *Governing Governance: A Formal Framework for Analysing Institutional Design and Enactment Governance*
2016-42 Spyros Martzoukos (UVA), *Combinatorial and Compositional Aspects of Bilingual Aligned Corpora*
2016-43 Saskia Koldijk (RUN), *Context-Aware Support for Stress Self-Management: From Theory to Practice*

2016-44 Thibault Sellam (UVA), *Automatic Assistants for Database Exploration*
2016-45 Bram van de Laar (UT), *Experiencing Brain-Computer Interface Control*
2016-46 Jorge Gallego Perez (UT), *Robots to Make you Happy*
2016-47 Christina Weber (UL), *Real-time foresight - Preparedness for dynamic innovation networks*
2016-48 Tanja Buttler (TUD), *Collecting Lessons Learned*
2016-49 Gleb Polevoy (TUD), *Participation and Interaction in Projects. A Game-Theoretic Analysis*
2016-50 Yan Wang (UVT), *The Bridge of Dreams: Towards a Method for Operational Performance Alignment in IT-enabled Service Supply Chains*

====
2017
====

2017-01 Jan-Jaap Oerlemans (UL), *Investigating Cybercrime*
2017-02 Sjoerd Timmer (UU), *Designing and Understanding Forensic Bayesian Networks using Argumentation*
2017-03 Daniël Telgen (UU), *Grid Manufacturing, A Cyber-Physical Approach with Autonomous Products and Reconfigurable Manufacturing Machines*