TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

Eeva Järvenpää

Capability-based Adaptation of Production Systems in a Changing Environment

Julkaisu 1082 • Publication 1082

Tampere 2012

Tampereen teknillinen yliopisto. Julkaisu 1082
Tampere University of Technology. Publication 1082

Eeva Järvenpää

Capability-based Adaptation of Production Systems in a Changing Environment

Thesis for the degree of Doctor of Science in Technology to be presented with due permission for public examination and criticism in Festia Building, Auditorium Pieni Sali 1, at Tampere University of Technology, on the 19th of November 2012, at 12 noon.

Tampereen teknillinen yliopisto - Tampere University of Technology
Tampere 2012

ISBN 978-952-15-2928-3 (printed)
ISBN 978-952-15-2940-5 (PDF)
ISSN 1459-2045

ABSTRACT

Today's production systems have to cope with volatile production environments characterized by frequently changing customer requirements, an increasing number of product variants, small batch sizes, short product life-cycles, the rapid emergence of new technical solutions and increasing regulatory requirements aimed at sustainable manufacturing. These constantly changing requirements call for adaptive and rapidly responding production systems that can adjust to the required changes in processing functions, production capacity and the distribution of the orders. This adaptation is required on the physical, logical and parametric levels.

Such adaptivity cannot be achieved without intelligent methodologies, information models and tools to facilitate the adaptation planning and reactive adaptation of the systems. In industry it has been recognized that, because of the often expensive and inefficient adaptation process, companies rarely decide to adapt their production lines. This is mainly due to a lack of sufficient information and documentation about the capabilities of the current system and its lifecycle, as well as a lack of detailed methods for planning the adaptation, which makes it impossible to accurately estimate its scale and cost. Currently, the adaptation of production systems is in practice a human driven process, which relies strongly on the expertise and tacit knowledge of the system integrators or the end-user of the system.

This thesis develops a capability-based, computer-aided adaptation methodology, which supports both the human-controlled adaptation planning and the dynamic reactive adaptation of production systems. The methodology consists of three main elements. The first element is the adaptation schema, which illustrates the activities and information flows involved in the overall adaptation planning process and the resources used to support the planning. The adaptation schema forms the backbone of the methodology, guiding the use of other developed elements during both the adaptation planning and reactive adaptation. The second element, which is actually the core of the developed methodology, is the formal ontological resource description used to describe the resources based on their capabilities. The overall resource description utilizes a capability model, which divides the capabilities into simple and combined capabilities. The resources are assigned the simple capabilities they possess. When multiple resources are co-operating, their combined capability can be reasoned out based on the associations defined in the capability model. The adaptation methodology is based on the capability-based matching of product requirements and available system capabilities in the context of the adaptation process. Thus, the third main element developed in this thesis is the framework and rules for performing this capability matching. The approach allows automatic information filtering and the generation of system configuration scenarios for the given requirements, thus facilitating the rapid allocation of resources and the adaptation of systems. Human intelligence is used to validate the automatically-generated scenarios and to select the best one, based on the desired criteria.

Based on these results, an approach to evaluating the compatibility of an existing production system with different product requirements has been formulated. This approach evaluates the impact any changes in these requirements may have on the production system. The impact of the changes is illustrated in the form of compatibility graphs, which enable comparison between different product scenarios in terms of the effort required to implement the system adaptation, and the extent to which the current system can be utilized to meet the new requirements. It thus aids in making decisions regarding product and production strategies and adaptation.

PREFACE

When I started my doctoral studies five years ago, I had no idea of the challenging journey I was about to take. During that journey, there have been moments when I felt hopeless and the completion of the thesis project seemed more like utopian dream than a realistically attainable goal. However, in the end I managed to pull everything together and the results of all that hard work are laid before you here. The project has been a true learning experience, teaching me not only about the topic I studied and how to become a researcher, but also about myself as a person. I could never have reached this point alone, and therefore I want to express my sincere gratitude to all those people who have stood by me and helped and motivated me along the way.

First, I want to thank all my present and former colleagues in the Department of Production Engineering of Tampere University of Technology for the friendly and instructive discussions around and beyond the thesis topic. In particular, I am grateful to my team leader, Dr. Minna Lanz, who has guided me throughout the whole journey, as well as to the present and former team members, Pasi Luostarinen, Fernando Garcia, Roberto Rodriguez and Timo Kallela, who patiently discussed my ideas with me and performed the programming tasks needed in order to validate them. Riku, Nikke, Timo, Asser, and all the rest of you who sit around the department's coffee table, thank you for putting up with my moments of anxiety and for sharing many pleasant chats with me. I could never have managed this without you all!

I wish to express my special thanks to my supervisor, Professor Reijo Tuokko, for giving me the freedom to choose my own research path and for his encouragement and inspiring words as I trod my chosen path. I am also very grateful to my external evaluators, Professor Michel Cotsaftis from Ecole Centrale d'Electronique, France and Professor Tullio Tolio from Politecnico di Milano, Italy for their highly valuable comments and suggestions, which have helped me to improve the quality of this work. I would also like to thank Adrian Benfield for reviewing the language of the thesis.

I have been fortunate enough to carry out my doctoral research as a graduate school student in the Graduate School of Concurrent Engineering led by Professor Erno Keskinen. I am most fortunate to have been given this opportunity and regard it as a major enabling factor in the success of this work. During the graduate school seminars I was able to develop my scientific thinking, obtain valuable peer-support from the other students and to meet people who will remain friends for the rest of my life.

My special thanks and hugs go to my family, my mother, sister, brother, grandma, auntie and cousin, for always being proud of me and letting me to be myself and to forget the researcher identity every now and then. Thanks are also due to my friends for their support throughout the project, and especially for those moments when they succeeded in making me forget the whole thesis. Finally, I wish to express my love and gratitude to my beloved Henkka for being there for me, supporting me in the stressful writing phase of this thesis and for sharing many enjoyable moments with me.

Tampere, 5th of October 2012

Eeva Järvenpää

ABBREVIATIONS AND ACRONYMS

ABAS	Actor-based Assembly System
ADT	Axiomatic Design Theory
AI	Artificial Intelligence
BMS	Bionic Manufacturing System
CAD	Computer Aided Design
CAM	Computer Aided Manufacturing
CAReP	Computer Aided Reuse Planning
CARP	Computer Aided Reconfiguration Planning
CE	Concurrent Engineering
CIP	Common Industrial Protocol
CNC	Computer Numerical Control
CoBASA	Coalition Based Approach for Shopfloor Agility
CO2PE!	Cooperative Effort on Process Emissions in Manufacturing -initiative
CSM	Competitive Sustainable Manufacturing
DB	Database
DeMO Tool	Decision Making and Ordering Tool
DFMA	Design for Manufacturing and Assembly
DIN	Deutches Institut für Normung (German Institute for Standardization)
DiMS	Distributed Manufacturing System
DL	Description Logics
DMS	Dedicated Manufacturing System
DOF	Degrees of Freedom
DP	Design Parameter
DSL	Domain Specific Language
EAS	Evolvable Assembly System
EU	European Union
EUPASS	Evolvable Ultra-Precision Assembly SystemS –project
FBS	Function-Behavior-Structure Framework
FMEA	Failure Mode Effect Analysis
FMS	Flexible Manufacturing System
FoV	Field of View
FP6	Framework Programme 6
FR	Functional Requirement
FrMS	Fractal Manufacturing System
GA	Genetic Algorithm
GUI	Graphical User Interface
HMS	Holonic Manufacturing System
ICT	Information and Communications Technology
ID	Identifier
IDEF0	Integrated DEFinition for Function Modelling
ISO	International Organization for Standardization
KB	Knowledge Base
KIPPcolla	Knowledge Intensive Product and Production Management from Concept to Re-cycle in Virtual Collaborative Environment –project

MAS	Multi-agent System
MES	Manufacturing Execution System
MPMS	Multiple Part Manufacturing System
MSDL	Manufacturing Service Description Language
MTBF	Mean Time between Failure
MTTR	Mean Time to Repair
NIST	National Institute of Standards and Technology
OWL	Web Ontology Language
PiSA	Flexible Assembly Systems through Workplace-Sharing and Time-Sharing Human-Machine Cooperation –project
PNG	Portable Network Graphics
Pro-FMA	Feature recognition tool
RDF	Resource Description Framework
RMS	Reconfigurable Manufacturing System
RPC	Remote Procedure Call
SADT	Structured Analysis and Design Technique
SIARAS	Skill-based Inspection and Assembly for Reconfigurable Automation Systems -project
SOA	Service Oriented Architecture
SOAP	Simple Open Access Protocol
SO-EAS	Self-Organizing Evolvable Assembly System
STEP	Standard for Exchange of Product model data
SPARQL	SPARQL Protocol and RDF Query Language
TS	Tabu Search
TUT	Tampere University of Technology
UI	User Interface
UML	Universal Modelling Language
UMRM	Unified Manufacturing Resource Model
URL	Uniform Resource Locator
USB	Universal Serial Bus
VDI	Verein Deutscher Ingenieure (Association of German Engineers)
VRML	Virtual Reality Modelling Language
WLAN	Wireless Local Area Network
X3D	eXtensive 3D, successor of VRML format
XML	eXtensive Mark-up Language

TABLE OF CONTENTS

1. INTRODUCTION

1.1. Industrial challenges and motivation

Nowadays manufacturing companies have to cope with several critical issues, such as frequently changing customer requirements, short product lifecycles, an increasing number of product variants, small batch sizes, varying volumes, stringent quality requirements, the rapid emergence of new technical solutions and increasing regulatory requirements aimed at sustainable manufacturing. Customers demand customized products at the price of mass-produced ones, and with fast delivery. Reporting on a study of German industry carried out by Wildemann in 2009, Zäh et al. (2011) pointed out that the number of product variants increased by between 500% and 700% between 1980 and 2002. Yet, over the same period, the production volumes remained the same, or even decreased to 85% of their original values (Zäh et al. 2011). This turbulent production environment calls for agile, adaptive and rapidly responding production systems that can adjust to the required changes in production capacity, processing functions and the distribution of orders. As defined by Koren and Shpitalni (2010), the responsiveness of a production system refers to the speed at which the system can meet changing business goals and produce new product models. They also stated that while responsiveness is not yet regarded as being as important as cost and quality, it is fast becoming a new strategic goal for manufacturing enterprises (Koren & Shpitalni 2010).

The Europe-wide strategic goal of Competitive Sustainable Manufacturing (CSM) calls for the re-use and adaptivity of production systems. Sustainable development consists of three structural pillars, namely society, environment, and economy. It involves operational aspects, such as the consumption of resources, the environmental impact and economic performance of the operations, the products, the workforce, social justice, community development, and so on (Jovane et al. 2009). The adaptation, and thus the re-use, of production systems can have a positive economic and ecological impact through the higher utilization of the resources of the existing system itself. Adaptation prolongs the lifetime of production systems and allows the potential of the existing resources to be utilized more effectively. Thus, it supports the sustainability paradigm, both from the economic and ecological perspectives.

Figure 1 compares the lifecycle of consumer goods with the lifetime of major components in the production system. It shows that resources and production systems can last much longer than the products they produce. This fact clearly indicates the potential for adaptation, and the consequent re-use of the production systems. The potential advantages of adaptation, such as savings in investment costs and production loss during the changeover of the product, faster ramp-up, and faster response to the customer requirements are clearly recognized within industry. However, in many cases, in the automotive industry, for example, the most common practice when the car model changes is still to disassemble the whole assembly line and build a new one from scratch.

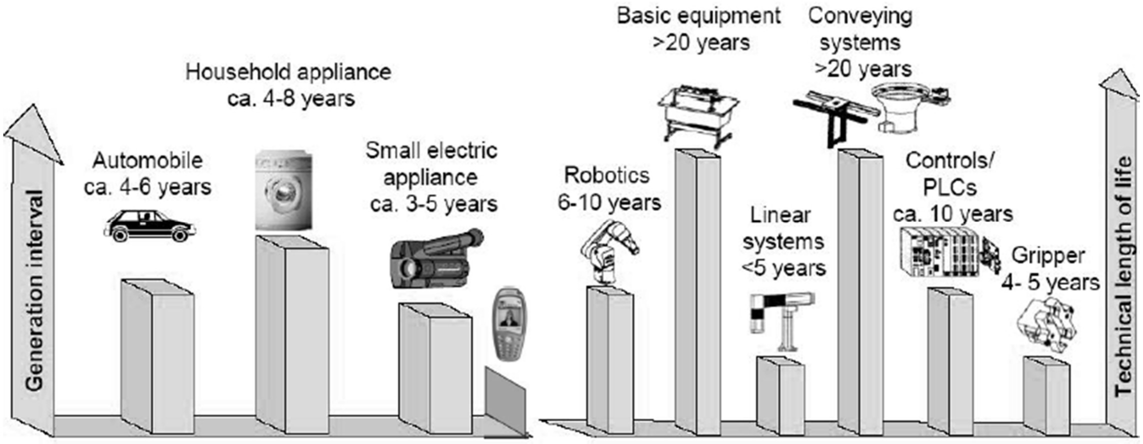


Figure 1. Product lifecycle versus system component lifecycle, modified from (Slama 2002).

Any change in the product or production requirements usually leads to changes in the production system that has to meet these requirements. These changes can be accommodated through the system's built-in flexibility or through reconfiguration. It is inevitable that changes incur costs and take time. Therefore, it would be beneficial not only to be able to adapt the existing system to the new requirements, but also to be able to estimate the scale of the required change (adaptation). This would then give an indication of the capital investment needed, other costs, and the timescale, which information could then be used to choose and prioritize between different product scenarios which require change.

Based on research carried out during the FP6 European Integrated Project called PiSA (Flexible Assembly Systems Through Workplace-Sharing and Time-Sharing Human-Machine Cooperation) by Fleschutz et al. (2008; 2009) and Harms et al. (2008; 2009) it was noted that companies rarely decide to reconfigure their assembly lines. The survey of selected European end-user companies and system integrators, which was conducted during the project, showed that both the producers and users of assembly equipment had little experience of the process of re-using existing equipment. Further problems were caused by inadequacies in the design and preparation of the production equipment, a lack of life-cycle data and monitoring concepts, insufficient information management and no holistic planning concepts for the adaptation and re-use of existing production facilities (Fleschutz et al. 2008; Harms et al. 2009.) Without up-to-date information about the capabilities and lifecycle of existing resources, it is practically impossible to estimate the scale and cost of any needed adaptation.

A cost analysis of two real re-use cases in the PiSA-project showed that the potential for cost reduction through the re-use of production components and systems was between 30 and 80 %. However, accurate cost estimates are difficult to make. For example, the value of an old system can only be approximated based on the effort required to remanufacture or retool it, and this is difficult to predict without the benefit of experience. Because there is no clear financial evaluation of the impact of re-use, any calculations depend strongly on vague assumptions. According to the interviews carried out during the project, the main cost drivers in the re-use of an assembly system are the adaptation processes needed to meet the required functional capabilities. Risks involved in the re-use of existing equipment arise through the accumulation of unforeseen adaptation processes. (Fleschutz et al. 2008.)

In order to ensure the economic and technical success of adapting an existing production system, information and knowledge management is essential. The information needed to evaluate the compatibility of the existing system with the new product requirements, the required modifications and consequently the effort and expense of the adaptation, is based on extensive knowledge from different sources in various fields of expertise. These are, for example, product design, process planning, system design and integration, operation and maintenance, as well as usage history and prior adaptation experiences. The management of such information in industry is not currently sufficient to generate reliable plans for adaptation, let alone dynamic reactive adaptation. What makes the management of this information difficult is its rapidly evolving nature. Another problem is the poor interoperability between the systems used to create, save, manage and utilize this information. Thirdly, traditional hierarchical information management, decision making and operation control systems are based on traditional stand-alone concepts that are optimized towards achieving their goals in static environments in which behaviour can be predicted. However, nowadays a production environment is anything but static. These traditional systems, designed for mass production, cannot cope with frequently changing orders and production volumes, or the production of customized solutions on batch size one.

1.2. Scientific and technical challenges

According to Rahimifard and Weston (2009), in order to be agile and adaptive, the production system needs to be able to meet the changing market requirements in terms of product characteristics and quantities. To do this, the production system needs an inherent ability to facilitate continual and timely change in its structure and in its functional operations. Structure refers to the way in which the functional building blocks of a production system are assembled to form a holistic, interoperable system, while the term function describes the abilities of the building blocks or the production system as a whole to realize a defined purpose. (Rahimifard & Weston 2009.)

Different manufacturing paradigms have been proposed in recent years to overcome the challenges relating to responsiveness and adaptivity. Flexible manufacturing systems (FMS) are designed to meet a wide variety of requirements without any physical modifications to the system structure (Koren 2006; Terkaj et al. 2009a). Reconfigurable manufacturing systems (RMS) also aim to meet these requirements by offering the rapid adjustment of production capacity and functionality in response to new circumstances, through rearrangement or change in both the system’s structure and the hardware and software components (ElMaraghy 2006; ElMaraghy 2009; Koren et al. 1999; Mehrabi et al. 2000). Agent-based and holonic systems take a more dynamic approach to dealing with the changeability requirements through the provision of self-organizing capabilities. Agents and holons are generally characterised as distributed, autonomous entities capable of intelligent behaviour and interaction with both their environment and other agents in order to achieve a particular goal (Monostori et al. 2006).

Most research into adaptive systems has focused on static adaptation, in which physical changes are made to the system “offline” when the system is not running. The agent-based and holonic approaches are often aimed at dynamic adaptation, in which logical or parametric changes happen “online”, while the system is running. Whereas static adaptation is usually based on planning, either by a human expert or through automatic planning methods, dynamic adaptation requires self-organizing capabilities from the system, because the system needs to react to changes in its environment. As stated by Westkämper (2006), self-organization, self-optimization, target-orientation and self-control are all characteristic features of the future structures of factories and production networks. The realization of these requirements calls for new methods and solutions that would drastically reduce the time and effort put into planning and implementing the alterations in a factory, such as plug and play interfaces, modern information and communication technologies, simulations and new planning methods. (Westkämper 2006.) Regardless of whether the adaptation is static or dynamic, or whether it occurs on a physical, logical or parametric level, intelligent methods and tools are needed to support the adaptation. In this dissertation the word 'adaptation' is used to cover all these different levels, which will be discussed in more detail in Chapter 3.1.2.

Previous research on reconfigurable and adaptive systems has concentrated on hardware and control technologies, as well as the physical structure of such systems (Bi & Zhang 2001a,b; A. I. Dashchenko 2006; ElMaraghy 2009; Martinez Lastra 2004; inter alia). Significant steps towards modular assembly equipment and standardized hardware and control interfaces have been made by, for example, the European Union-funded project called EUPASS (Evolvable Ultra-Precision Assembly SystemS) (EUPASS 2009). According to Ferreira et al. (2010), the modular architecture paradigm for new production systems, which focuses on the clear functional decoupling of equipment module functionalities and the use of standardized interfaces to promote interchangeability, presents the possibility for developing automated adaptation methods. However, the standardization of hardware and software interfaces is not in itself enough to enable the rapid adaptation of a production system. Efficient methodologies, tools and information models are needed to support humans in the adaptation planning process, and also to enable reactive adaptation to take place while the system is running. These methodologies should also take into consideration the fact that most of the systems on today's factory floors are still not modular and they lack standardized interfaces. However, it is still desirable that they can produce different products in different volumes, and that production in a changing environment can be planned and controlled. Regardless of whether the system is characterized as "reconfigurable" or not, it will still have to undergo some changes during its lifetime.

Most of the proposed methods in the field of physical adaptation design have focused on structural configurations at the single machine level (Bi & Zhang 2001b; O.A. Dashchenko 2006; Jang et al. 2008; Moon 2006; Tang 2005; inter alia), while the systems themselves have usually been designed intuitively (Bi et al. 2008). There are no holistic integrated methodologies which simultaneously take into account the old production system, its layout, the reference system architecture (i.e. the type of system that can be implemented), new product requirements, resource capabilities, resource history, the lifecycle and condition of the equipment, and of course economy and efficiency. As noted in, for example, both the PiSA and EUPASS projects (Fleschutz et al. 2008; Harms et al. 2009; Ferreira et al. 2010), the adaptation planning of production systems today is, in reality, a human driven process, which relies strongly on the expertise and tacit knowledge of the system integrators or the end-user of the system. Even though this process may result in feasible system configurations, it seldom follows a systematic approach to planning the adaptation, which means that the solutions are neither replicable nor transparent. Such a time-consuming planning process may produce expensive plans, making frequent system adaptations unfeasible.

To conclude this introduction to the scientific and technical challenges involved in adaptation, and to encapsulate the driving force behind this thesis, it can be stated that there are three main reasons why the adaptation of production systems, both static and dynamic, has been so rare to date. These are: 1) the lack of extensive methodologies and tools to plan the adaptation at the whole production system level; 2) the lack of adequate information models and tools for gathering and managing information, in order to support both the adaptation planning and reactive adaptation. For example, information about the capabilities and lifecycle of current systems is lacking from existing models; 3) the lack of distributed, decentralized, modular ICT and control architectures which support the self-organizing capabilities of the systems. This thesis contributes to the first two points, while the case studies used in the work also provide an insight into the third point.

2. RESEARCH DESCRIPTION

2.1. Introduction to the research problem

Chapter 1.2 discussed the scientific and technical problems which hinder the adaptation of production systems in the industry today. As was stated, a proficient mechanical design and an intelligent control system design are not enough in themselves to support the efficient adaptation of production systems in a changing environment. An intelligent methodology, including tools and information models, is needed to support both rapid adaptation planning controlled by human experts and the dynamic reactive adaptation of the systems.

This thesis aims to provide solutions for the following two problems as stated in Chapter 1.2:

Problem 1: Lack of extensive methodologies and tools to plan the adaptation on the production system level;

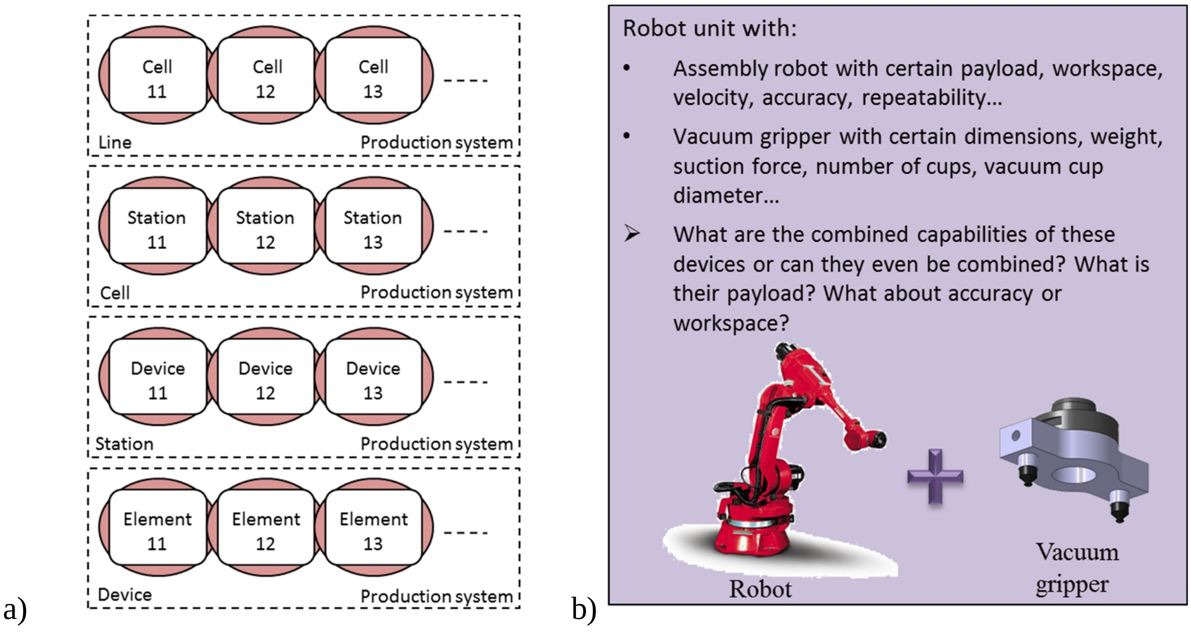
Problem 2: Lack of sufficient information models, as well as tools for information capture and management, to support the adaptation planning and reactive adaptation.

Two types of change stimuli for the production system have to be considered. These are: lifecycle dynamics relating to long-term changes, such as new installations or the replacement of old equipment (static); and, run-time dynamics relating to short-term changes which need to be solved while the system is running, such as re-routing and re-scheduling (dynamic). So, both of these approaches to adaptation, and any combination of the two, should be supported by the new methodology:

1. Static adaptation - Adaptation happening “offline” when the system is not running. Static adaptation needs planning, either by humans or through automatic planning methods.
2. Dynamic adaptation - Adaptation happening “online” when the system is running. Dynamic adaptation requires the system to have self-organizing capabilities, in other words, the system needs to autonomously react to any changes in its environment.

In the context of production system adaptation, resource models represent the key factor. Based on the literature review presented in Chapter 0, a critical factor for the computer-aided adaptation of a production system is efficient resource models, which provide the information needed for equipment selection and system integration. It is obvious that a digital representation of the capabilities of a system and its components would significantly ease the design of the system and its integration, adaptation and operation. It would allow an automatic procedure to find suitable system components and to build alternative scenarios during the early phases of adaptation planning. This requires a formalized representation of the functional capabilities of the resources, along with any other relevant properties they may have. However, presenting the simple capabilities of individual resources is not enough when adapting complete production systems. Therefore, what is also needed is a way to describe the combined capabilities of multiple co-operating resources.

The adaptation planning problem deals with the different levels of a system simultaneously, as shown in Figure 2a. When adapting a production system for a new requirement, it is preferable to view the system (and its capabilities) from the top, rather than to consider each individual resource in the system separately. For example, there may be stations that could be utilized as they are without any physical changes. However, in order to identify them, a formal description of the combined capabilities of the station composed of multiple devices is needed. According to Ueda et al. (2001), the design of production systems follows emergent synthesis, wherein the local interactions between the artifacts of the system dictate the global behaviour through bottom-up development to achieve the purpose of the whole system. On the other hand, the global behaviour results in new constraints on the behaviour of the elements in the system (Ueda et al. 2001). Due to these local interactions, the combined behavior of resources is something other than the sum of the behavior of each individual resource. Consider, for example, the problem illustrated in Figure 2b.



Robot unit with:

- Assembly robot with certain payload, workspace, velocity, accuracy, repeatability…
- Vacuum gripper with certain dimensions, weight, suction force, number of cups, vacuum cup diameter…

➢ What are the combined capabilities of these devices or can they even be combined? What is their payload? What about accuracy or workspace?

Figure 2. a) Partonomy of different system levels. b) Example of the combined capability problem.

Several approaches and models to describe the resources exist, as will be discussed in Chapters 3.3 and 3.4. However, these approaches don't support such modelling of resource combinations and their combined capabilities, at least not in sufficient detail. This is a considerable problem for adaptation planning. Therefore, much of the emphasis in this work is on solving this capability-based resource description problem. As stated in Chapter 1.2, many earlier adaptation methods have concentrated on structural configurations at the single machine level. This thesis aims to concentrate on the whole system level. Therefore, the desired granularity in the resource description is from the line to the device level, rather than going down to the level of the individual elements.

2.2. Formulation of the research objectives and questions

The high-level objective of the thesis: Development of a methodology to support computer-aided adaptation of production systems in a changing environment.

In the context of this work, the term 'production system' refers to discrete manufacturing and assembly systems. The term 'changing environment' refers to a production environment which is characterised by changing requirements in terms of product variants, changing product models, fluctuating volumes, changing characteristics (capabilities) of the production system components and the integration of new technologies. The aim is to reduce the planning work traditionally done by humans by developing computerized solutions, especially for information management, handling, and filtering. In particular, the aim is to facilitate the creation of feasible adaptation solution scenarios and hence to reduce the need for manual planning work. The term ‘feasible’ here refers to a situation where the system is compatible with the requirements, but not necessarily optimal. The methodology concentrates on evaluating the changes the system needs in terms of hardware. The cost and investment calculations are beyond the scope of this thesis. However, once these hardware changes are identified, this would also provide a foundation for estimating the investment needed.

Based on the literature survey a following ‘working statement’ was adopted as a premise for this work to guide the development of the methodology:

> *The adaptation of production systems to changing requirements, both static, human-centric adaptation as well as dynamic reactive adaptation, can be eased by a formal capability-based resource description and framework and rules for matching the resource capabilities against product requirements in the context of the adaptation process.*

The backbone for such an adaptation methodology is a suitable model (schema) that can capture all the relevant aspects of the adaptation process, including the activities, the information and process flows during the adaptation planning, and reactive adaptation. The main focus of the work is on the development of a formal resource description model, which will enable the automatic matching of product requirements with system capabilities and is, therefore, the most fundamental element of the methodology. In addition, a framework and rules are needed to enable this matching. In order to complement the methodology, it is also desirable to have an approach to evaluating the impact of the changes in product requirements on the production system under various product scenarios. Figure 3 represents an overview of the problem definition of this thesis with the two supported viewpoints.

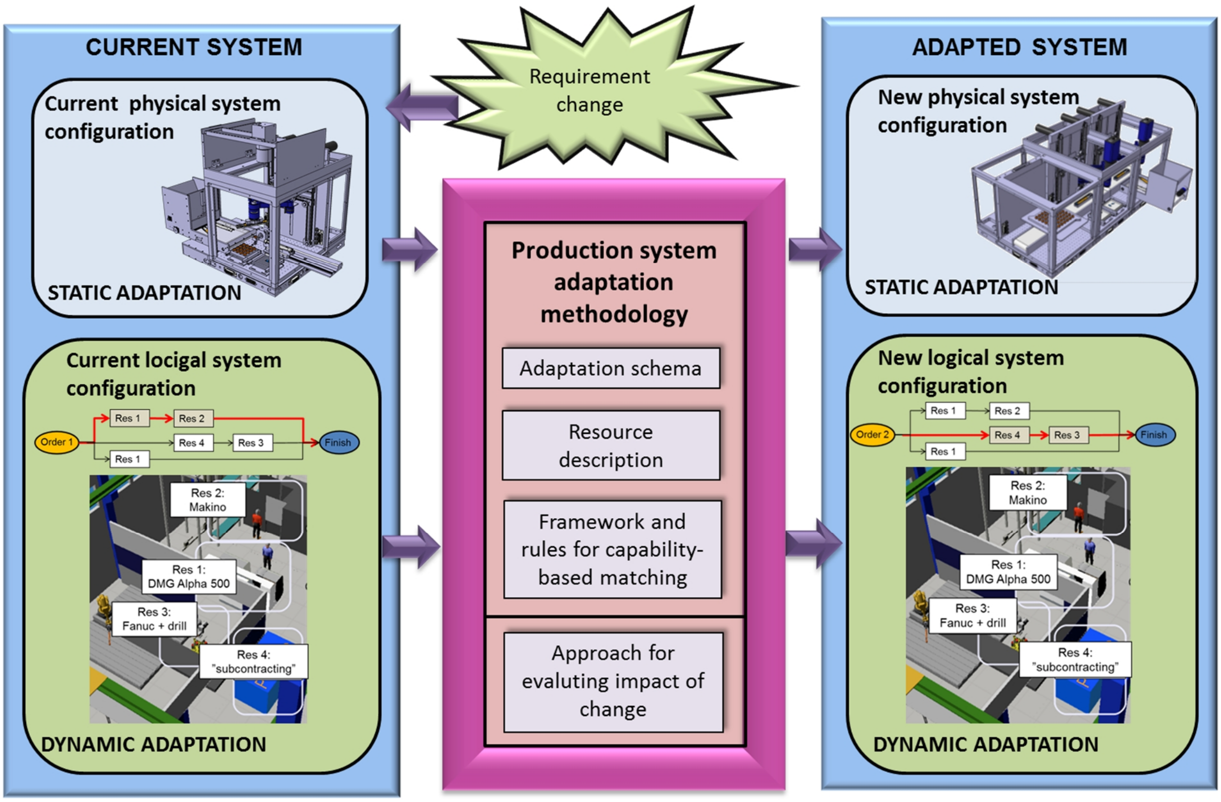


Figure 3. Problem definition overview.

The high-level objective presented at the beginning of this chapter is divided into the following sub-objectives:

Sub-objective 1: Adaptation schema
The first sub-objective is to define an adaptation schema, which illustrates the overall adaptation planning process and activities, as well as the information requirements and flows to support both human-controlled adaptation planning and reactive adaptation.

Sub-objective 2: Resource description model based on capabilities
The second, yet the most important of these sub-objectives, is to develop a resource description model which can capture the relevant resource characteristics, and support both the adaptation planning and the reactive adaptation of the production systems. The resource description model should enable the automatic matching of product requirements and resource capabilities and provide the basis for further automatic reasoning. It should also enable the description and management of the capabilities of multiple co-operating resources.

Sub-objective 3: Framework and rules for capability-based matching in adaptation
The third sub-objective is to define a framework and rules for matching the product requirements against the system capabilities during adaptation planning and reactive adaptation.

Sub-objective 4: Preliminary approach to evaluating the impact of change
The fourth sub-objective is to devise a preliminary approach to evaluating the impact of the changes in the product requirements on the production system. This approach should be based on the results of the implementation of the previous three sub-objectives.

Based on the above objectives, the following associated research questions were formulated:

Research question 1: How should the overall adaptation planning process be presented?
What method can be used to present the overall adaptation planning process? What are the steps in the adaptation planning process? What kind of information resources are needed during the adaptation planning? How is the information created and what is the information flow between the different activities?

Research question 2: How should the resource information be presented so that it allows the automatic matching of product requirements against resource capabilities?
What kind of resource characteristics need to be included in the resource model to support adaptation planning and reactive adaptation? How should the capabilities of the resources be modelled? How should the combined capabilities of multiple co-operating resources be represented? How should the link between the simple and combined capabilities be managed?

Research question 3: How should the resource capabilities be matched with the product requirements during adaptation planning and reactive adaptation?
What is the process and what steps need to be taken when matching the product requirements to the resource capabilities in the adaptation context? What kind of rules are needed for this matching?

Research question 4: How can the impact of changes in the product requirements on the production system be evaluated?
How should the solutions developed for the previous research questions be utilized for such an evaluation? How should different product scenarios which require change be compared?

2.3. Research methodology

Instead of trying to describe and understand the reality behind the current situation, the aim of this work is to solve a practical problem and thus improve the current situation. Therefore, the research methodology has been adopted from design science. In this thesis, Design Research Methodology (DRM) (Blessing & Chakrabarti 2009) is utilized as the basic framework for clarifying the research problem, formulating the proposed solution and analysing the results. The DRM is divided into the following stages and respective means and outcomes:

1) Research Clarification (RC)
   At this stage, the goals of the research are formulated through an analysis of the relevant literature.
2) Descriptive Study I (DS-I)
   At this stage, there is an extended analysis and review of the literature and empirical data may be collected to gain a deeper understanding of the research problem and influential factors.
3) Prescriptive Study (PS)
   At this stage, all the experience, knowledge and assumptions about the problem are synthesized into a proposed solution.
4) Descriptive Study II (PS-II)
   At this stage, the proposed solution is analysed and evaluated against the defined goals using empirical data.

According to Blessing & Chakrabarti (2009), DRM shouldn't be interpreted as a set of stages and supporting methods to be executed rigidly and linearly. Instead, it should be used flexibly and should allow multiple iterations. For example, Descriptive Study I provides the means to further clarify the research problem from its initial formulation.

In order to develop a solution proposal, the classical method of knowledge-based system development is utilised. It consists of the following phases (Buchanan & Duda 1983):

- Identification: identifying the characteristics of the problem and setting goals;
- Conceptualization: making the key concepts and their relations explicit;
- Formalization: presenting the identified concepts in formal language;
- Implementation: presenting the knowledge from the previous step in a system shell;
- Testing: utilizing sample cases to test the system and identify any weaknesses;
- Revision: revising the previous steps based on the test results.

Utilizing these two above-mentioned methodological frameworks, the objectives of the thesis were approached using the following steps and associated research methods. The steps are divided into three categories: awareness of the problem and understanding the topic; development of the proposed solution; evaluation and conclusions.

Awareness of the problem and understanding the topic (RC, DS-I)

Step 1: Identifying the research problem and objectives
The research started with an extensive literature survey, focusing mainly on manufacturing- and production-related journals and refereed conference articles and compilations. Bibliometric methods, such as citation analysis and co-citation coupling were used to assemble the relevant literature. The main goal of the survey was to outline the theoretical framework and requirements

for the study, and to recognize the state of the art in the field. The objectives for the research were defined based on the literature survey.

The development of a proposed solution (PS)

Step 2: Adaptation schema definition

Based on the understanding gained from the previous step, it was recognised that a fundamental enabler for efficient adaptation is an understanding of the activities and information flows which characterize the adaptation process. Therefore, the actual development work started by defining the initial adaptation schema that illustrate the activities, information flows, resources and controls used during adaptation planning and reactive adaptation. This forms the backbone of the adaptation methodology. The schema was defined based on the literature and the author's own experience. Several iterations were made after steps 3 and 4 before the final adaptation schema was formulated with IDEF0 (Integrated DEFinition for Function Modelling) activity diagrams.

Step 3: Development of a resource description model based on capabilities

The third step developed an ontological resource description model which describes both the resources and their capabilities, and the combined capabilities of multiple co-operating resources. The work started by defining the requirements for the resource description model, and then tackling the resource description problem using a case-based methodology, utilizing the TUT-machining laboratory and the TUT-micro-factory environments as cases in point. Together, these two case environments provided a relatively broad view of different production resources and processes, thereby facilitating the development of a generally applicable model. Datasheets of the resources were used as primary information about the resource characteristics when generating the model. The existing Core Ontology by Lanz (2010) was used as a basis for the knowledge modelling, and in this thesis, this was reinforced with information about the resource capabilities. In order to tackle the combined capability problem, the upper level capabilities (combined capabilities) were divided into lower level capabilities (simple capabilities) using the functional decomposition method. The capabilities of the two case environments (and other similar ones) were included in the instantiated capability model, containing currently over 90 capability instances. A Capability Editor tool was developed to allow the capabilities to be modelled to the Core Ontology.

Step 4: Development of framework and rules for capability-based matching

The fourth step started with a definition of the overall framework for matching the product requirements against the production system capabilities. After that, the capability taxonomy was defined to enable the high-level mapping of resources and product requirements. Several existing process descriptions and standards were used as a basis for the development of the taxonomy. The rules and rule base were developed on a conceptual level in order to facilitate detailed capability matching. The development of the rule base began by defining what kind of rules are needed, how they should be organized and how they can be used in the adaptation planning and the reactive adaptation, after which a number of rules were written and implemented. In order to take into account the specific nature of adaptation, the rule base consists not only of rules relating to capability matching, but also adaptation-related rules and guidelines, which may be used as part of the capability matching and in the selection of the final configuration. Again, the case-based approach was used in the development of these rules.

Evaluation and conclusions (DS-II)

Step 5: Validation of the developed methodology with realistic case studies
In the fifth step, the developed adaptation methodology and the models and concepts contained therein were validated with realistic case studies. The first case, implemented in the TUT-machining laboratory, represents a holonic production environment where the adaptation takes place dynamically through reaction and self-organization. The second case study, modelled in the TUT-micro-factory, represents a more traditional production system where the adaptation is based on human-controlled planning. The resulting resource description was tested and validated by describing the capabilities of these two case environments, which consisted of real production resources. Altogether about 65 individual resources were described with the model. A number of the capability-matching rules were also implemented to validate the capability-matching framework.

Step 6: Evaluation of the proposed methodology
In the final step, the methodology which had been developed was evaluated against defined objectives, and the conclusions of the study were formulated.

2.4. Limitations and assumptions

As the broad focus of this thesis touches on a number of different research areas, certain limitations and assumptions need to be stated. Firstly, in the scope of this thesis, the product is assumed to be a known input, even though it is undeniable that there is an iterative, multi-directional and dynamic relation between products, processes and production systems. This thesis starts with the assumption that the product is defined, and no modifications are made to the product based on the system specifications. Naturally, the feedback loop from production to the product designer's table is crucial, but product design itself is not considered in this thesis. Nor does the methodology developed in this thesis consider the production control side.

Secondly, the aim is not to develop an optimal resource description covering all the possible resource characteristics, but to develop a resource model which provides enough expressiveness to meet the requirements of the two case studies. For example, the developed resource description is not intended to model the kinematics of the machines, because there are multiple other models available for modelling those, such as virtual simulation models. Thirdly, the framework and rules which were developed for capability matching are not intended to completely replace human expertise from this reasoning phase. Instead, the approach developed here is intended to support human expertise through the automatic generation of scenarios whose feasibility can be assessed by the human expert in order to make an informed final decision.

Fourthly, the aim of the adaptation methodology is not to optimize the adaptation plans, but to find feasible solutions. The thesis concentrates on the adaptation of systems in a changing environment where the system itself, and the requirements placed on it, are constantly changing and cannot be foreseen. Optimizing this kind of system is not practical for two reasons. Firstly, all the information that would be needed to obtain a reliable solution may not be available. Secondly, in a changing environment, the solution that is optimal at one point in time may already be obsolete at the next point.

Finally, it has to be noted that although the literature review carried out for this thesis aims to summarize the research activities of other researchers investigating similar research questions, it must be accepted that there may be other implementations and methods published in unrecognised conferences or poorly indexed publications which have not been considered here.

2.5. Thesis outline

The thesis is structured as shown in Figure 4.

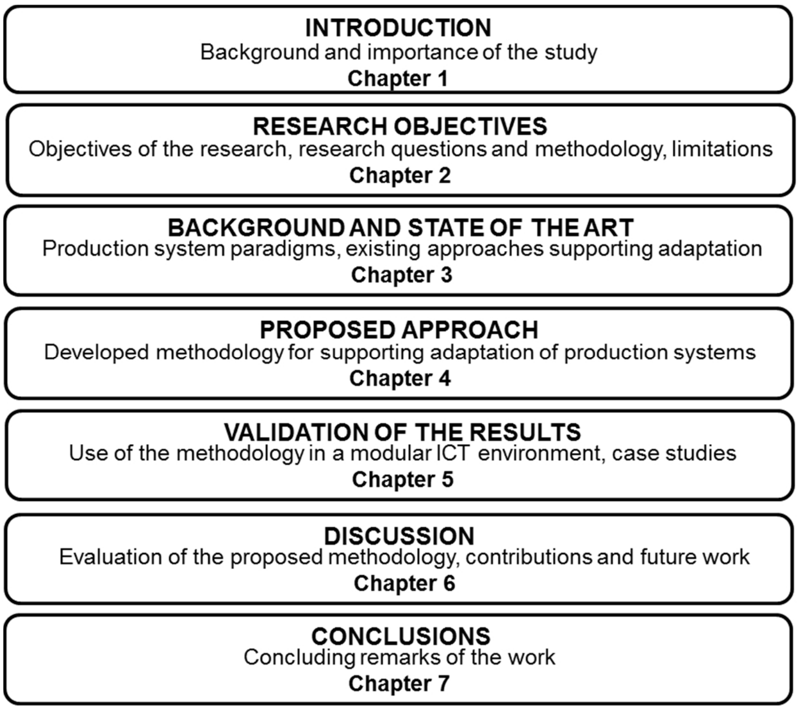

INTRODUCTION
Background and importance of the study
Chapter 1
RESEARCH OBJECTIVES
Objectives of the research, research questions and methodology, limitations
Chapter 2
BACKGROUND AND STATE OF THE ART
Production system paradigms, existing approaches supporting adaptation
Chapter 3
PROPOSED APPROACH
Developed methodology for supporting adaptation of production systems
Chapter 4
VALIDATION OF THE RESULTS
Use of the methodology in a modular ICT environment, case studies
Chapter 5
DISCUSSION
Evaluation of the proposed methodology, contributions and future work
Chapter 6
CONCLUSIONS
Concluding remarks of the work
Chapter 7

Figure 4. Outline of the thesis.

Chapter 1 presents the introduction to the research topic by discussing the challenges related to production system adaptation from both the industrial, and the scientific and technological viewpoints. Chapter 2 states the research objectives and questions. It also discusses the selected research methodology and the limitations and assumptions of the research. The literature and state-of-the-art review is covered in Chapter 3. It reviews existing and emerging production system paradigms and the current approaches to production system adaptation, and their limitations. The developed methodology for computer-aided production system adaptation, including the proposed solutions for the sub-objectives, is presented in Chapter 4. Chapter 5 describes the implementation of the developed methodology in a modular ICT-environment. The main focus of this chapter is to validate the results in the context of two practical case studies, representing two different aspects of adaptation, namely static human-controlled adaptation planning and dynamic reactive adaptation. In Chapter 6, the developed methodology is evaluated against the set objectives, and there is a discussion of the contribution this research has made to the problem, and the direction for further study. Finally, the conclusions of the study are presented in Chapter 7.

3. LITERATURE REVIEW

In recent years, much research has been initiated in the area of novel production paradigms and approaches that can cope with the changing requirements and conditions in the production sector. Three main research directions can be identified: 1) new production system paradigms around adaptive manufacturing; 2) the development of modular system components with standardized interfaces; 3) enhancing the flexibility and adaptivity of production environments through the use of intelligent, autonomous, real-time control-systems. What have been less thoroughly researched are methodologies which facilitate adaptation planning and dynamic reactive adaptation. These are the main points of interest in this literature review.

There are a lot of different methods for production system design. However, most of these methods focus on the design of a new production system, while relatively few researchers have studied how an existing production system can be adapted in order to meet changes in the requirements for the system. Methodologies for generating structural (re-)configurations at the single machine level do exist, but proper extensive methodologies and tools for the adaptation of complete production systems are not so well established.

This section aims to furnish the background to the development of the adaptation methodology by surveying the 'state of the art' in the field. First, different production system paradigms will be discussed in Chapter 3.1. Chapter 3.2 discusses traditional approaches to system design and reconfiguration as an extension of traditional system design. Chapter 3.3 then reviews the existing approaches of a number of researchers to production system adaptation, and finally, in Chapter 3.4, the information models supporting adaptation are reviewed.

3.1. Production system paradigms

This chapter aims to provide a theoretical framework for the study. First some traditional production system paradigms are briefly discussed, followed by a discussion of adaptive production systems, adaptation and related terminology. After that, complex adaptive systems and systems intelligence will be discussed, followed by the emerging evolutionary system approaches, which utilize the theory of complex systems. Finally, this chapter will finish by comparing static and changing systems based on the theory presented in the previous chapters, and lay down the foundation for the developments followed in this thesis.

3.1.1. Traditional production system paradigms

The three major types of traditionally-recognized manufacturing systems are dedicated, flexible and reconfigurable manufacturing systems. The characteristics of these system types are discussed briefly below.

Dedicated manufacturing systems (DMS)
Dedicated manufacturing systems are based on inexpensive, fixed automation that produces a company's core products or parts at high volumes and over long periods of time. These dedicated systems are usually designed to produce a single part at a high production rate, which is achieved by utilizing multiple tools simultaneously. (Koren & Shpitalni 2010.) Dedicated systems are tailored to

certain static production requirements and thus their performance is robust and the initial capital cost is low (Landers et al. 2006).

Flexible manufacturing systems (FMS)
Flexible manufacturing systems are designed to produce a variety of products with a changeable mix on the same system with short change-over times. They are very often comprised of general-purpose CNC-machines and other programmable forms of automation. The production rate of a flexible system is low compared to that of a dedicated system. Furthermore, due to the combination of high equipment costs and low productivity, the cost per part of using FMS is relatively high. (Koren 2006; Koren & Shpitalni 2010.) According to Landers et al. (2006) flexible manufacturing systems are not tailored to any specific requirement, but are designed to accommodate a wide range of changes in production requirements, and therefore they often contain excess capability, which results in unnecessary cost for the customers. In other words, the flexibility is built into the system and the capabilities of the system are pre-determined. For example, in the case of a multifunctional gripper, the gripper has got multiple fingers that are used or not, depending on the shape of the product to be handled. Furthermore, the capability limits of flexible systems are fixed and therefore they cannot cope with unforeseen requirements, which may exceed their built-in flexibility.

Reconfigurable manufacturing systems (RMS)
Reconfigurable manufacturing systems (RMS) aim to combine the high throughput of dedicated systems and the flexibility of flexible manufacturing systems. Instead of providing the general flexibility with built-in high functionality of FMS, reconfigurable systems aim to provide customized flexibility. (Koren & Shpitalni 2010.) The objective of RMS is to offer rapid adjustment of production capacity and functionality, in response to new circumstances, through the change or rearrangement of both its structure and its hardware and software components (Koren et al. 1999; Mehrabi et al. 2000). These components can be, for example, machines and conveyors in whole systems, mechanisms in individual machines, new sensors, or new control-algorithm software. The new circumstances requiring changes could be, for example, changing product demands, the production of a new product on an existing system, or the integration of new process technology into existing manufacturing systems (ElMaraghy 2006). As noted in the survey by Terkaj et al. (2009a), many systems simultaneously incorporate the characteristics of both flexible and reconfigurable systems.

Koren (2006) and Koren & Shpitalni (2010) stated that in order to achieve exact flexibility in response to fluctuation in demand, the following key characteristics must be taken into consideration when designing an RMS: modularity (components are modular), integrability (interfaces for rapid integration), customization (flexibility limited to part family), scalability (design for capacity changes), convertibility (design for functionality changes) and diagnosability (design for easy diagnosis). These are seen as the enablers for reconfiguration. Wiendahl et al. (2007) pointed out that, especially in the case of assembly systems, two more enablers should be added to this list. The first is mobility, which is important for reconfiguring single stations or modules in an assembly system, or even for moving the whole system to another location. The second enabler is the ability to upgrade or downgrade the degree of automation. (Wiendahl et al. 2007.)

3.1.2. Adaptive production systems and adaptation

Wiendahl and Heger (2004) identified five types of changeability for manufacturing systems: reconfigurability, changeoverability, flexibility, transformability and agility. Later, Wiendahl et al.

(2007) used changeability as a general term to characterise the ability of a system to economically accomplish early and foresighted adjustment of a factory's structures and processes at all levels in response to the change stimulus. Based on the literature around flexible, reconfigurable and adaptive manufacturing (ElMaraghy 2009; ElMaraghy 2006; Koren 2006; Mehrabi et al. 2000; Tolio & Valente 2006; inter alia), it is difficult to completely differentiate these concepts. Flexibility is often referred to as the ability to adapt to different requirements without physical changes to the system, whereas reconfigurability refers to the ability to change system components when new requirements arise (ElMaraghy 2006). However, these definitions can be used only if the boundary of the system is clearly defined. Tolio and Valente (2006) stated that, depending on the boundary, the type of the changeability can be interpreted as reconfigurability or as flexibility and therefore, it is not possible to define general statements for these characteristics. ElMaraghy (2006) divided a manufacturing system's reconfigurability into both physical and logical reconfiguration, touching on both definitions of flexibility and reconfigurability. Terkaj et al. (2009a) presented a framework for classifying these different flexibility-related forms and dimensions.

In order to avoid misunderstandings caused by the multiple definitions of reconfigurability, flexibility and so on, the term 'adaptation' is used in this work to cover both physical adaptation (reconfiguration) and logical adaptation, as seen in Figure 5. Besides those two types of adaptation, parametric adaptation is also included in this definition of the term. Such a definition is more relevant to the goal of this thesis, which is to support the adaptation of production systems in a changing environment without restricting itself to reconfigurable or flexible systems. Later in this thesis, the word "reconfiguration" may be used to specifically refer to physical adaptation.

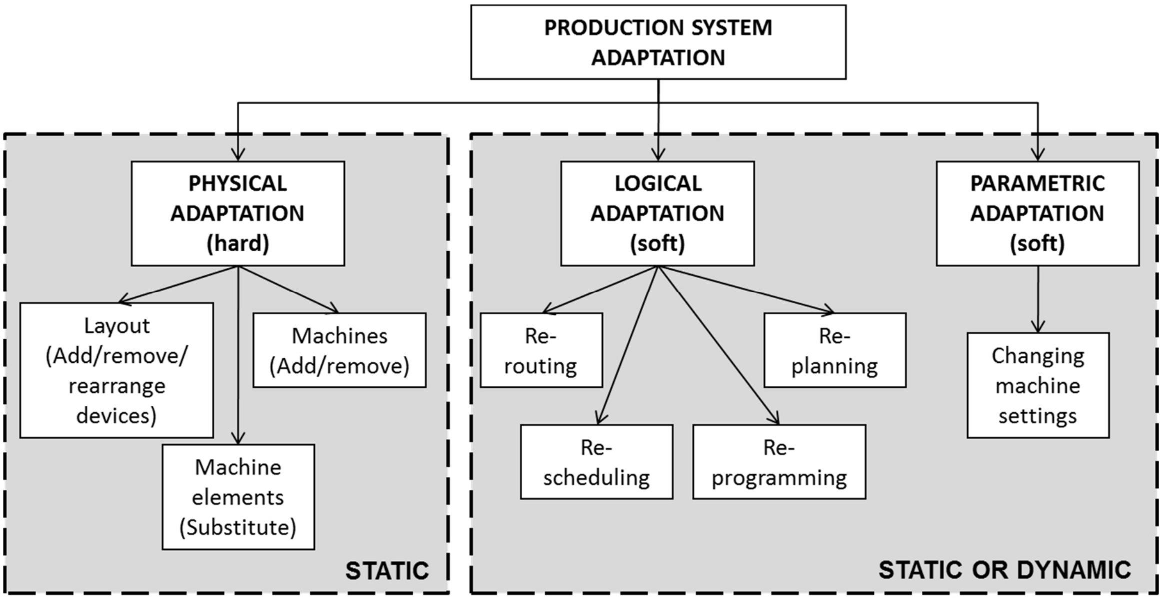

PRODUCTION SYSTEM ADAPTATION
PHYSICAL ADAPTATION (hard)
Layout (Add/remove/ rearrange devices)
Machines (Add/remove)
Machine elements (Substitute)
STATIC
LOGICAL ADAPTATION (soft)
Re-routing
Re-planning
Re-scheduling
Re-programming
PARAMETRIC ADAPTATION (soft)
Changing machine settings
STATIC OR DYNAMIC

Figure 5. Production system adaptivity, modified from (ElMaraghy 2006).

As Figure 5 shows, there are three basic types of adaptation which affect the configuration of the production system. According to Brehmer and Wang (1999) the configuration of a system refers to those aspects of a system that remain in a temporarily fixed state during a given time frame. Those fixed aspects may include three elements: a complete list of constituent components; the attributes and properties of each component; and the relationships between the components, e.g. spatial, temporal, functional, logical, etc. (Brehmer & Wang 1999). The three types of adaptation shown in Figure 5 are:

Type 1: Physical adaptation

- Physical adaptation means that physical changes are made to the system, such as changing the layout of the system, adding or removing machines or machine elements.
- Adapting the available capabilities.

Type 2: Logical adaptation

- During logical adaptation, no physical changes are made to the system, as any changes occur on a logical level. These can be achieved, for example, by changing the process sequence and re-routing or re-scheduling production.
- Adapting the utilization of available capabilities.

Type 3: Parametric adaptation

- Parametric adaptation doesn't include any physical or structural changes either, but refers to changing the adjustable machine parameters, such as the speed of a conveyor line.
- Adapting the behavior of available capabilities.

These three types of adaptation are not usually independent of each other. For example, physical changes in the production system often cause logical changes to the old configuration. However, the change requirement is usually only targeted to one type of adaptation at a time, so that any other types of changes follow naturally from the initial adaptation.

Adaptation can also be divided into static and dynamic adaptation. Static adaptation refers to changes in the system configuration during downtime of the system. Dynamic adaptation is the changes in the system configuration while the production system is in operation. These dynamic changes can involve either logical or parametric adaptation. Dynamic adaptation allows the production system to react to changes in its environment in real-time, for example, to recover from disturbances in the production line, and to self-organize itself to balance the production flow. While physical adaptation is usually done at the static level, both logical and parametric adaptation can be either dynamic or static. This means that logical and parametric changes can be executed either while the system is running or during its downtime.

3.1.3. Complex Adaptive Systems and Systems Intelligence

In recent years, production system research has started to adopt ideas from complex systems. According to Bourgine & Johanson (2006), complex adaptive systems are non-linear systems with many strongly-coupled degrees of freedoms. They are composed of multiple, interacting autonomous units, such as agents, actors or individuals having adaptive capabilities. The adaptive capability is manifested by the ability of these multiple-component systems to learn and evolve based on internal and external dynamic interactions. The components and organizational structures are able to react to their environment and feedback, and adapt their functions to novel conditions and tasks. (Bourgine & Johnson 2006.) According to Fryer (2010), the control of complex adaptive systems is highly dispersed and decentralized and they are characterized by constant, self-organizing behavior in order to find the best fit with the operating environment.

Complex systems are characterized by emergent phenomena that cannot be easily construed through a knowledge of their components alone. Complex systems usually consist of at least two different levels: the macro-level and the micro-level. Whereas the macro-level is concerned with the system as a whole, the micro-level views the system in terms of its local components. In multi-level complex systems, the higher-level system processes and behavior result from lower-level cooperative emergence, caused by interaction between the components. Similarly, lower-level system processes may be influenced, constrained or even determined by higher-level interactions. Due to the emergence resulting from communication and interaction between the components, the behavior and capability of the whole system is not just a straight-forward aggregate of the local components' behaviors and capabilities. (Ueda et al. 2001; Bourgine & Johnson 2006; Lanz et al. 2012.)

The concept of system intelligence is closely related to complex systems. Saarinen and Hämäläinen (2004) define system intelligence as "intelligent behaviour in the context of complex systems involving interaction and feedback". Any object with system intelligence can successfully and productively interact with the holistic feedback mechanisms of its environment. The object perceives itself as part of the whole, and takes into account the influence of the whole upon itself as well as its own influence upon the whole. The intelligent actions of the object are enabled by observing its own interdependence with its feedback-intensive environment. (Saarinen & Hämäläinen 2004.)

Saarinen and Hämäläinen (2010) summarized the key features of the "systems" of system intelligence (i.e. complex systems) collected from the relevant literature (Jervis 1998; Senge 1990; Ramage & Shipp 2009; Jackson 2003):

- The behaviour of the system displays features that cannot be obtained by summing up the behaviours of the isolated components, in other words the system can display emergence.
- The system has the ability to self-organize its components in order to create a new structure and, consequently, new behavior.
- In order to determine the overall behavior of the system, the relationships and interaction between parts, giving rise to patterns, regularities and complexity, are more important than an analysis of the properties of the individual parts in isolation.
- The systems are dynamic, showing changing states and behaviors during their lifecycle. These changes are often conceptualized in terms of functions or goals. Due to the non-linearity of the system, a change in one component may have an unanticipated effect on other components, and on the behavior of the system as a whole.
- The boundaries of these systems are re-definable, flexible and dependent on one's perspective, and are artificially created by humans for the sake of clarity and sanity.

Based on the Systems Intelligence approach, system adaptation is only possible through learning. Learning, on the other hand, requires observation and feedback (Bourgine & Johnson 2006). For an adaptive production system, feedback loops and an understanding of the feedback are essential. However, in the case of static adaptation, the feedback and its processing don't need to happen in real time. The learning within a production system can occur at either at the human or the system level, and this learning can result in adaptivity for any of the types presented in Figure 5, above.

Complex Systems science is closely related to the same phenomena in nature and society. Even though production systems cannot compete in complexity with the social or natural interactions of man or nature, they do have some characteristics of complex systems. Production systems are composed of numerous, individual and cooperating resources, which are connected to each other and have multi-lateral interactions. The global capability of the complete system is determined by the emergence of the combined capabilities from the capabilities of its individual resources. If we consider a production system as an ecosystem in which machines, people and computers co-operate, the complexity is clearly visible. Today's markets, which the production systems need to serve, are highly dynamic and unpredictable. The systems need to adapt themselves to these changing requirements. For this reason, it is seen that the study of complex systems could shed light on the current research into production systems. However, because the methods for solving the problems of complex systems are not yet mature enough, in this thesis the problem is treated rather as complicated than complex. On the other hand, this work will result an information model, which aids a system in becoming self-organising; one of the characteristics of complex systems.

3.1.4. Emerging evolutionary system paradigms in production

In order to cope with today's dynamic operating environment, and the resulting changes in requirements, several system paradigms which mimic the characteristics of natural and complex adaptive systems, such as different self-x capabilities, have been initiated. Some of these approaches will be discussed in the following chapters.

Fractal Factory

The fractal factory concept of Warnecke (1993) postulates that a manufacturing company is composed of small components, or fractal entities. As summarized by Tharumarajah et al. (1998), the three core internal features of fractals include: 1) self-organization, which implies freedom for the fractals to organize and execute tasks and to choose their own methods for solving problems, which include self-optimization enabling improvements in the process; 2) dynamics, whereby the fractals can adapt to influences from the environment without any formal organizational structure; 3) self-similarity, which can be understood to mean similarities in the goals for the various fractals, which conform to the objectives of each unit.

Bionic Manufacturing Systems

The bionic manufacturing systems (BMS) approach takes its inspiration from biological systems and proposes concepts for future manufacturing systems. Based on biologically inspired ideas, such as self-growth, self-organization, adaptation and evolution, BMS aims to deal with non-predeterministic changes in manufacturing environments. (Ueda et al. 1997; Ueda 2007.) As stated by Tharumarajah et al. (1998), a biological system is characterized by spontaneous autonomous behavior and social harmony within hierarchically ordered relationships. An example of a biological system is a cell, which is the basic unit from which the other parts of a biological system are composed. Cells can have different capabilities from each other, and they are capable of multiple operations. Translated into the manufacturing world, a manufacturing unit can be seen as a cell, as a building block for hierarchical control structures such as workstations, manufacturing cells, factories, business units or even production networks. In this kind of structure, the layers in the hierarchy support each other. The components in the system communicate and inform each other about their decisions. (Tharumarajah et al. 1998.)

Agent-based and holonic systems

According to Monostori et al. (2006) agent-based computation has revolutionized the building of intelligent and decentralized systems. The agent technologies are well suited for manufacturing domains where there are problems with uncertainty and temporal dynamics, information sharing and distributed operation, or the coordination and cooperation of autonomous entities. Agents are generally characterised as distributed, autonomous entities capable of intelligent behaviour, and interaction with their environment and other agents, in order to achieve a particular goal. Multi-agent systems are formed by a network of agents that interact and communicate with each other in a language which provides them with a common syntax and semantics. Depending on the context, an agent can be a human, an organization, a machine, a piece of software or any other actor in the system. According to a review of agent applications in the manufacturing domain, conducted by Monostori et al. (2006), agent-based approaches have been applied to engineering design, process planning, production planning and resource allocation, production scheduling and control, process control, monitoring and diagnosis, enterprise organization and integration, production in networks, and assembly and lifecycle management. (Monostori et al. 2006.)

Differing from the fully heterarchical agent-based systems, a holonic approach represents a transition between fully hierarchical and heterarchical systems (Monostori et al. 2006). The holonic concept was originally developed by the philosopher Arthur Koestler in order to explain the evolution of biological and social systems. The word “holon” is a combination of the Greek word “holos”, meaning whole, and the Greek suffix “on”, meaning particle or part. Therefore the term holon means something that is itself both a whole, and part of a greater whole. (Koestler 1967.) Translated into the manufacturing world, the concept behind a Holonic Manufacturing System (HMS) views the manufacturing system as one entity consisting of autonomous modules (holons) with distributed control. These holons are able to fulfill their own goals based on their own assessment of a situation, combined with their individual local knowledge, while simultaneously communicating and co-operating with other holons in order to meet the higher-level global goals. The holonic system is open. It has no explicit control system, but thanks to cooperation (common interfaces and negotiation) the holons may form a temporal, or permanent, federation, known as a holarchy. Thus, a holonic organization tries to combine the responsiveness and robustness of decentralized, network-like organizations with the stability and efficiency of hierarchical control architectures. The objective of holonic manufacturing systems has long been to transfer the benefits that holonic organizations provide for living organisms and societies to manufacturing, i.e., stability in the face of disturbances, adaptivity and flexibility in the face of change, and the efficient use of the available resources through self-organizational ability. (Giret & Botti 2004; Monostori et al. 2006.)

The following two examples of agent-based and holonic approaches involve utilizing the concepts of complex systems and system intelligence. The first is the Evolvable Assembly System (EAS) approach, which was proposed in 2002 and developed during the EUPASS project. The second approach, the Distributed Manufacturing System (DiMS) is at this stage more a conceptual framework than an actual implementation.

Evolvable Assembly System (EAS)

The aim of the EAS is to cope with unpredictable and changing production requirements by building evolvable capabilities into the production system. As stated by Onori et al. (2010), evolvability is not only the ability of system components to adapt to changing requirements, but also a characteristic, which assists the processes in becoming more self-x, which can stand for self-evolvable, self-reconfigurable, self-tuning, self-diagnosing and so on. The theoretical background behind the EAS concept is associated with systems theory and complex systems, such as complexity theory, artificial life, autonomic computing, agents, self-organization and emergence. (Onori et al. 2010.) EAS is characterized by distributed, multi-agent-based control, along with embedded intelligence at the component level, intelligent process-oriented modules with well-defined interfaces, reconfigurable modules with fine granularity, and open architecture. The technical and architectural aspects of EAS development are supported by a methodological framework which includes the EAS Reference Architecture describing the essential features of an evolvable assembly system. It specify the characteristics a system should have in order to be an evolvable system; Emplacement and Blue Print concept, which allows the description of a module's characteristics; the Skill concept, which represents the functional skills of the modules; and the EAS Ontology and knowledge model, which provides common concepts for different stakeholders and defines the relations between different concepts. (Semere et al. 2008; Onori et al. 2011.)

Distributed Manufacturing System (DiMS) framework

The DiMS (Distributed Manufacturing System) framework, developed by Nylund et al. (2008) and Salminen et al. (2009) at TUT is based on a holonic architecture in which the holons are independently communicating entities. In DiMS, the production environment is seen as dynamic and evolving, i.e. it's an open complex system, where the communication and decision making is based on the negotiation process between these entities. According to Nylund et al. (2011), the DiMS framework is a development method which supports manufacturing companies in decisions about how to improve their ability to adapt to changing requirements right from the earliest, conceptual, phases. It utilizes several different manufacturing paradigms, including holonic manufacturing systems (HMS), fractal manufacturing systems (FrMS), biological manufacturing systems (BMS), emergent synthesis, and service oriented architecture (SOA). (Nylund et al. 2011.)

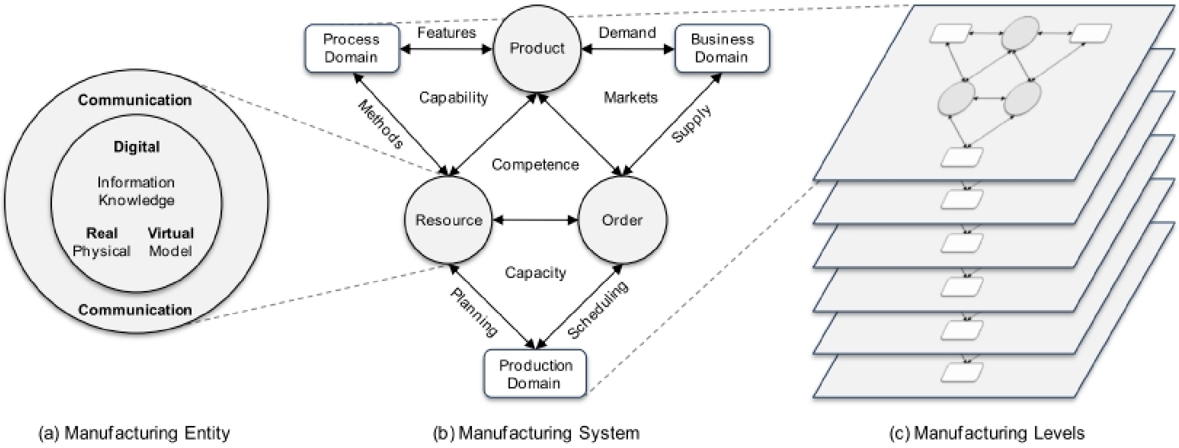


Figure 6. General representation of the DiMS system entity, structure and levels (Nylund et al. 2011).

As seen in Figure 6, the DiMS structure can be explained from three viewpoints: the internal structure of manufacturing entities; the total manufacturing system including the manufacturing entities and their related domains; and, the different fractal structuring levels of manufacturing. The entities have internal structures consisting of digital, virtual, and real parts. (Nylund et al. 2011.) The proposed structure is loosely based on the Product-Resource-Order-Staff (PROSA) reference architecture by Van Brussel et al. (1998), which describes a manufacturing system with three types of basic holons: resource holons, product holons and order holons. In the DiMS framework, the products represent what is offered to customers, the orders explain what the customers are actually ordering, and the resources are entities in the manufacturing system. The entities are connected with the process, production, and business domains. The process domain encompasses the manufacturing capabilities while the production domain is responsible for ensuring that there are enough resources to manufacture orders within given delivery times and in the required quantities. The business domain is responsible for markets. (Nylund et al. 2011.) DiMS is still largely only a conceptual framework.

However, there have been initial, partial implementations of it as described in Lanz et al. (2010b) and Järvenpää et al. (2011b; 2012a).

3.1.5. Static versus changing systems

This chapter compares static and evolving systems in terms of the theory presented in the previous chapters, wherein lies the reason why this thesis is not aimed at building an optimization or expert system for the adaptation planning problem. The fundamental reason for this is that this thesis is dealing with changing and unpredictable environments and systems, rather than with predictable ones. Optimization models and methodologies for analysing and optimizing deterministic systems can be found. In these kinds of systems the goal is known, the system is known and there is no randomness in the system, i.e. the deterministic model will always produce the same output from a given starting condition or initial state.

Traditionally, system design has operated under the assumption that the system is static and closed to external inputs. Current design and planning systems don't support any other method. However, real world systems, such as production systems, are not static and deterministic closed-world systems, but rather need to evolve in response to unpredictable, external inputs. Therefore the problem should be approached from a stochastic, or open-system viewpoint, as is suggested by the evolutionary system paradigms presented in Chapter 3.1.4. Table 1 summarizes the characteristics of static closed-world systems and changing open-world systems with information collected and adapted from the multiple sources discussed in Chapter 3.1.4.

Table 1. Comparison between static closed and dynamic open systems.

Characteristics of a static closed system	Characteristics of a changing open system
• Environment the system (ob)serves is assumed to be static. • The system is a closed world system and doesn't take input from outside world. • Because the environment is static, the system can be optimized. • The system is hierarchical and extremely rigid. • Only pre-defined interactions between the system parts exist. • The behavior of the system is predictable.	• Environment the system (ob)serves changes unpredictably. • The system is an open world system, which accepts input from the outside world. • The system adapts in response to the inputs in order to survive in its environment. • The system, as well as the environment, requires feedback from actions. • The system structure is not rigid, but adaptive. • Interrelated system elements work towards a common goal; there are multiple causes, effects and interactions. • Due to multiple interactions and random factors coming from the environment, the behavior of the system can not be predicted.

This thesis concentrates on the adaptation of systems in a turbulent environment where the system itself and the requirements placed upon it change unpredictably. Optimizing this kind of system is not feasible because of the three main issues discussed below.

The first difficulty, when working with industrial problems, is that any definition of the cost function, or a precise problem statement, is inextricably linked with the design process. In many cases, the designer's preferences cannot be articulated accurately or may not even be known. The adaptation planning problem, like most industrial problems, is not a single-objective optimization problem, but has multiple, often conflicting, objectives. Multi-objective optimization refers to the process of systematically optimizing a number of objective functions simultaneously. The primary goal of multi-objective optimization is to model a decision maker's preferences (ordering or prioritising objectives and goals). (Marler & Arora 2004.) With this kind of multiple-objective optimisation problem, it is difficult to talk about the optimum, since there is no common agreement on what the optimum is. The adaptation planning problem has a complicated structure precisely because of its multiple components, including the selection of devices, maximization of the use of existing system components, system availability, scheduling, efficiency, cost, and so on. A criterion which is valued in a specific situation may change from case to case, and new criteria may emerge. It is therefore impossible to formulate a generic and reusable objective function.

Secondly, this thesis deals with dynamic, changing environments, in which not only the environment but also the system are both constantly changing. A solution that is optimal at one point in time may already be obsolete at the next point, because the prevailing conditions have changed. As can be seen from Table 1, it is, in practice, impossible to optimize these kinds of dynamic open systems, simply because their behavior is not predictable. Therefore, there is no point in wasting time trying to define highly accurate objective functions and problem statements, if they can't provide the correct solutions anyway.

Thirdly, it is highly probable that all the information required for a reliable solution is not available. Adaptation planning needs information from a variety of dispersed sources, and such information is often insufficient. The planning process has to make do with the information that is available. Consequently, it is not feasible to try to build an expert system which can make adaptation decisions automatically. An expert system is a knowledge-based system designed to encapsulate the accumulated human expertise in a particular, specialized domain, which is to say that it mimics the capabilities of human experts (Hopgood 2001). Expert systems can only treat problems within narrow limits, rather than more broadly relevant and realistic issues. Another important issue is that expert systems are no use if the data they are working with is incorrect, or even merely inaccurate, as this may result in misleading solutions (Slack et al. 2004). Adaptation planning requires information from multiple sources and is usually highly dependent on the context from which it emanates. The context-information cannot yet be handled by the current, computerized, information management systems. In summary, building an expert system is not feasible because it would need to make decisions based on information which doesn't exist, which would lead to highly unreliable solutions.

3.2. Traditional system design and physical adaptation

According to Koren et al. (1999), reconfiguration, i.e. physical adaptation, is a natural extension of modular system design, with the primary difference that it needs to take the existing system into consideration. Unlike traditional system design, in reconfiguration design the system has already been built and only needs a few modifications. It is not economically viable to design the new system from scratch, even though the new system would probably be better optimised for the new application. This is especially true when batch sizes are small (or even single units). Therefore, traditional methods need to be modified before they can be used for reconfiguration design.

Mehrabi et al. (2000) indicated that there are two levels in the reconfiguration design problem, the system level and the machine level. First, the reconfigurable system is designed at the system level. The product features are related to the required processing units and layout. In other words, it is the properties of the product that define what kind of processes and operations are required, and, therefore, what kinds of machine types are needed. In the light of these system level specifications, the optimal design for the machines is defined in detail at the machine level. (Mehrabi et al. 2000.) Bi et al. (2008) stated that there are basically two methods for designing a new configuration for an existing system. The first approach is to design a new system solution from scratch, and then to compare it with the existing system in order to establish what changes are required. This is not a very practical method, however, because it can lead to major, yet unnecessary changes. The second approach is to start with the original specification for the existing system, and to change that until it meets the new requirements. (Bi et al. 2008.)

Which methodology is used for configuration and reconfiguration depends on the 'coupling' nature of the system. Systems can be classified as uncoupled, loosely-coupled or strongly-coupled. In uncoupled systems, one physical module satisfies one functional requirement. In strongly-coupled systems, one physical module doesn't solely satisfy one functional requirement, but rather, all the design variables have to be considered together before deciding whether the proposed configuration fulfils its requirements. (Bi et al. 2008.)

The algorithmic design approach, proposed by Pahl and Beitz (1996), defines a procedure that needs to be carried out during the design process. Pahl and Beitz divided the system design process into four main phases: planning and clarifying the task, conceptual design, embodiment design and detail design. They stated that both 'problem decomposition and synthesis' approaches and 'hierarchical design' approaches are aimed at dealing with the inherent complexity in large system engineering. Their approach is to decompose the overall design problem into sub-problems until a satisfactory level of simplicity and clarity has been reached. At this level, sub-solution alternatives can be defined. If these solutions are found to satisfy the original problem definitions, they are combined until the full system has been defined. (Pahl & Beitz 1996.) At this point, the system solution has to be validated against the original problem.

According to Avgoustinov (2007), in reality it is very rare for a solution to match the problem exactly. While the decomposition stage constitutes the analysis phase of problem-solving, combining the solutions and matching them to the original problem constitute the synthesis phase, as shown in Figure 7 (Avgoustinov 2007).

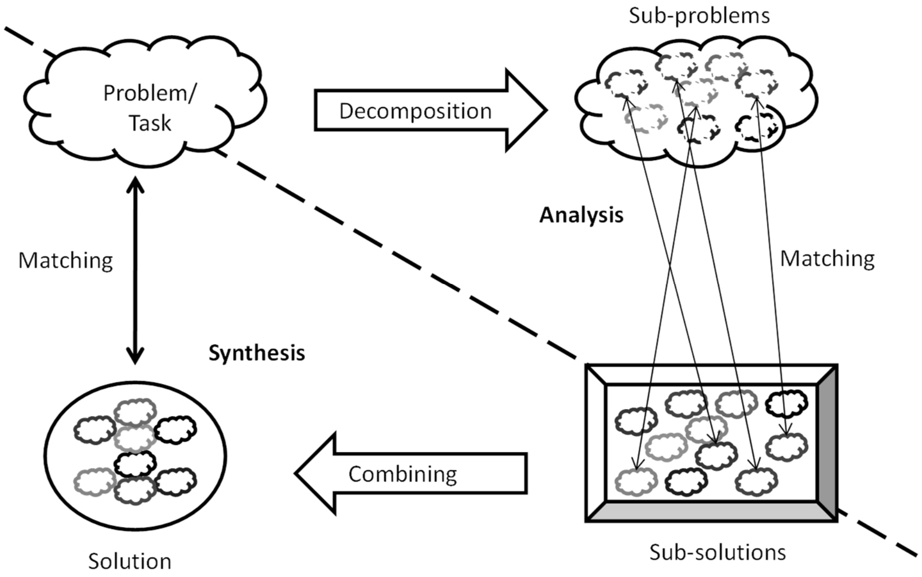


Figure 7. Problem solving by decomposition.

According to Avgoustinov (2007), the sizes of the sub-solutions indicate their granularity. Clearly, a coarse-grained solution would be put together faster but match the problem less closely than a fine-grained solution. (Avgoustinov 2007.) In terms of production system design, this means that the more finely-grained the modules are, the better the solution will match the original production system design problem. However, finely-grained modules would drastically increase the time needed for the design phase, as well as the complexity of the design process as a whole. It would also probably result in greater changes to the production system during any subsequent adaptation phase. Therefore, when starting adaptation planning, it is preferable to view the system as a whole entity, from the top, than to consider each individual resource on the system, especially if the optimality of the system is not crucial.

Suh (1998) presented an Axiomatic Design Theory (ADT) for engineering system design. The axiomatic design approach defines domain models and rules, which are used to generate a new design. The axiomatic design method focuses on the generation of functional requirements (FRs) and the selection of design parameters (DPs) to fulfil the requirements. The FR's and DP's are connected by means of design matrices, as shown in Equation (1). The binary elements of the design matrix, expressed as X's and 0's, indicate the presence or absence of a relationship between a DP and its associated FR. (Suh 1998.)

$$\begin{Bmatrix} FR_1 \\ FR_2 \\ FR_3 \end{Bmatrix} = \begin{bmatrix} X & 0 & 0 \\ X & X & 0 \\ X & 0 & X \end{bmatrix} \begin{Bmatrix} DP_1 \\ DP_2 \\ DP_3 \end{Bmatrix} \tag{1}$$

The two axioms, i.e. the Independence and Information Axioms, are used to select the best set of possible design parameters. The first axiom states that when multiple FRs exist, the design solution must be such that each FR can be satisfied without affecting the other FRs. Once this objective has been achieved, the design matrix will be diagonal, as each DP will affect only its associated FR, and there will be no coupling between the off-diagonal elements. The information axiom states that simpler designs are better. (Suh 1998.)

Bi and Zhang (2001a) proposed a general strategy for determining the configuration of a modular system based on ADT. They noted that the function-means diagram used in traditional, modularity-based design is the same as the FR-DP diagram used in ADT. If a given modular architecture satisfies the Independence Axiom, i.e. one physical module satisfies one functional requirement, and if the functional decomposition is consistent with the modular architecture, the determination of a modular configuration can be organized hierarchically. (Bi & Zhang 2001a.) Nevertheless, with complex production systems there is rarely a one-to-one correspondence between the FRs and the DPs. Figure 8 shows an FR-DP diagram for modular robots, in which a number of 'many-to-one' and 'many-to-many' connections between FRs and DPs can be seen.

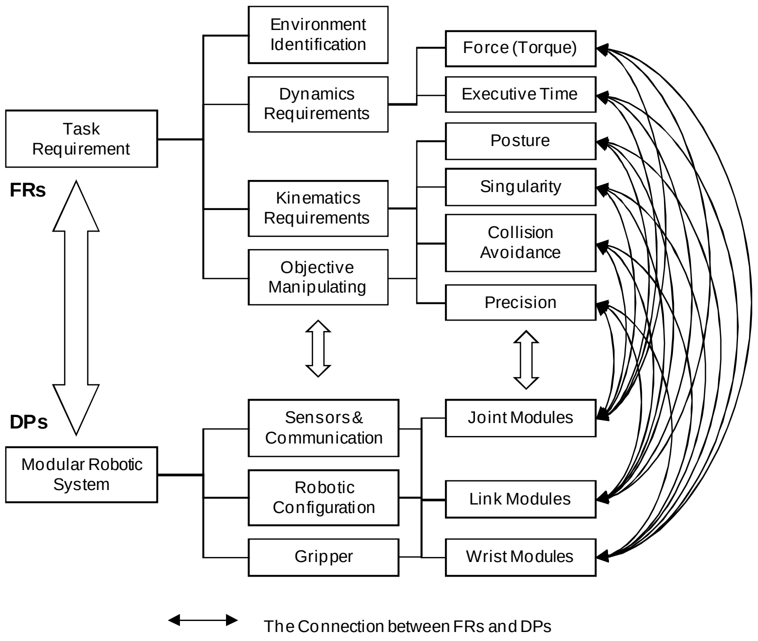


Figure 8. Coupled design of modular robotic configuration (Bi & Zhang 2001a).

The lowest granularity level considered in this thesis problem is at the device level, meaning that the devices are not divided into smaller elements, such as joints. For example, a production system may consist of many different types of devices, such as machine tools, robots, grippers, feeders, conveyors, and so on, which themselves, when divided into elements, are complex and thus strongly-coupled. However, the complete system itself can be seen as loosely-coupled, because each component in the system corresponds to one main functional requirement (e.g. feeder to feed, robot to pick and place, conveyor to transfer), yet all those components together affect the overall performance of the production system. According to Bi et al. (2008), multiple methods can be applied to loosely-coupled modular system design problems, such as feature-based methods (Perremans 1996), modular-based methods (Tsai & Wang 1999), the combinatorial synthesis method (Levin 2002), entity-based methods (Hong & Hong 1998) and case-based methods (Watson 1999).

Rampersad (1994) developed a method that simultaneously considers the design of products, processes and systems, as opposed to most other methods, which use product design and processes as information sources, rather than actively changing them. Vos (2001) proposed an assembly system development method based on modular assembly equipment. First, he developed a method for module specification. Second, a configuration method for module selection was created. He identified the matching of process requirements with the capabilities of the modules as one of the most critical problems, and used operations as a basis for the matching. (Vos 2001.)

The work of De Lit and Delchambre (2003) concentrated on assembly line design. They divided the assembly line design into the following categories, which have here been adapted with small modifications to cover both the assembly and manufacturing domains:

- Manufacturing and assembly planning – Manufacturing and assembly plan shows how the product can be manufactured or assembled, specifying the order of the operations that have to be carried out in order to produce the final product.
- Resource planning – Resource planning aims to select an appropriate means of production in order to perform the operations defined in the previous phase, while meeting the production

volumes set by marketing. The resource planning aims to attribute operations to workstations (WSs) and select resources that will perform the different operations and tasks.

- Line balancing – Line balancing aims to balance the workload between stations.
- Line layout – The line layout design deals with the precise design of the manufacturing or assembly line, and defines the exact location of the resources on the line.

Lohse (2006) stated that, especially for the design of assembly systems, the decomposition process involves a simultaneous detailing of the assembly process and grouping the necessary process steps into equipment requirements. The ability to achieve the required set of assembly process steps is the first criterion for equipment selection. Only then are the specific behaviors of the various alternative solutions taken into consideration in order to optimize the performance of the overall system. Again, the combined process capabilities of the chosen equipment solutions drive the synthesis and validation. (Lohse 2006.)

Resource planning and equipment selection plays an important role in both the original system configuration and its reconfiguration. Over the last few years, methods based on object-oriented technology have emerged for selecting the tools and machines in the system design. For example, Usher and Fernandes (1999) proposed a systematic method for identifying and ranking tool alternatives based on the production costs and time. They developed production time and cost factors to rank selected tools, and used the object-oriented approach to handle a large number of alternative tool sets. (Usher & Fernandes 1999.) Grabowik and Knosala (2003) presented a method for representing knowledge in computer-aided production planning. Their approach introduced an object-oriented method that presents the design features as individual objects within a hierarchical class structure. They applied the proposed method on an expert system that searches through the knowledge base to find technical solutions, e.g. machine tools for particular applications. (Grabowik & Knosala 2003.)

Rekiek et al. (2002) recognized that most automated methods in assembly system design concentrate on balancing the assembly line and assigning the tasks to the workstations. These methods consider the equipment as a given, and do not consider any alternative equipment. They found a few methods for tackling the resource planning problem, but the drawback with these is that they assume that all the alternative types of equipment are capable of performing the required assembly operations, and merely try to optimize the selection according to certain criteria such as cost or cycle-time. (Rekiek et al. 2002.) They don’t consider the process requirements dictated by the product, and are therefore unsuitable for the type of adaptation planning under consideration in this study. This thesis particularly concentrates on the selection of suitable resources, rather than balancing them, and therefore the existing approaches are not of much use. More recent approaches to resource selection in the adaptation context are presented in Chapters 3.3 and 3.4.

3.3. Existing approaches to production system adaptation

This chapter reviews a number of existing methodologies that support adaptation from different viewpoints. The first three sections are purely related to reconfiguration. They are divided into generic frameworks for reconfiguration planning, artificial intelligence-based approaches for reconfiguration planning, and knowledge-based approaches for reconfiguration planning. The two following sections also touch on the parametric and logical aspects of adaptation and are divided into capability-based approaches to adaptation and agent-based approaches to adaptation. In this study, the existing approaches are analysed in terms of how much they have to contribute to the original research problem. Finally, their shortcomings are summarised in Chapter 3.3.6, which justifies the need for new research. For the sake of clarity, the terminology of the original references is used when referring to these approaches, and so, in most cases, the word "reconfiguration" is used instead of "physical adaptation".

3.3.1. Reconfiguration planning frameworks

Deif and ElMaraghy (2006) presented an open, mixed architecture both for the design of reconfigurable manufacturing systems, and for controlling this design process. The architecture, shown in Figure 9, is divided into three layers. The first layer is dedicated to capturing the market demand, while the second concentrates on generating and selecting the best configuration that satisfies this demand. The third layer is concerned with the actual physical arrangements of the selected configuration. Each design layer is controlled by different performance measurements that reflect the strategic objectives of the reconfigurable manufacturing system. Even though the methodology presented here concentrates on the design of the initial configuration, the reconfiguration planning is also taken into account. (Deif & ElMaraghy 2006.) The architecture gives a comprehensive explanation of the reconfiguration process and visualizes the different areas that need to be developed in such systems. It gives an overall view of the RMS design process, but doesn't provide detailed means for performing the individual design steps. The methodology is not aimed at automating any of the steps in the reconfiguration planning process, and therefore lacks the models and mechanisms which would enable automatic reasoning.

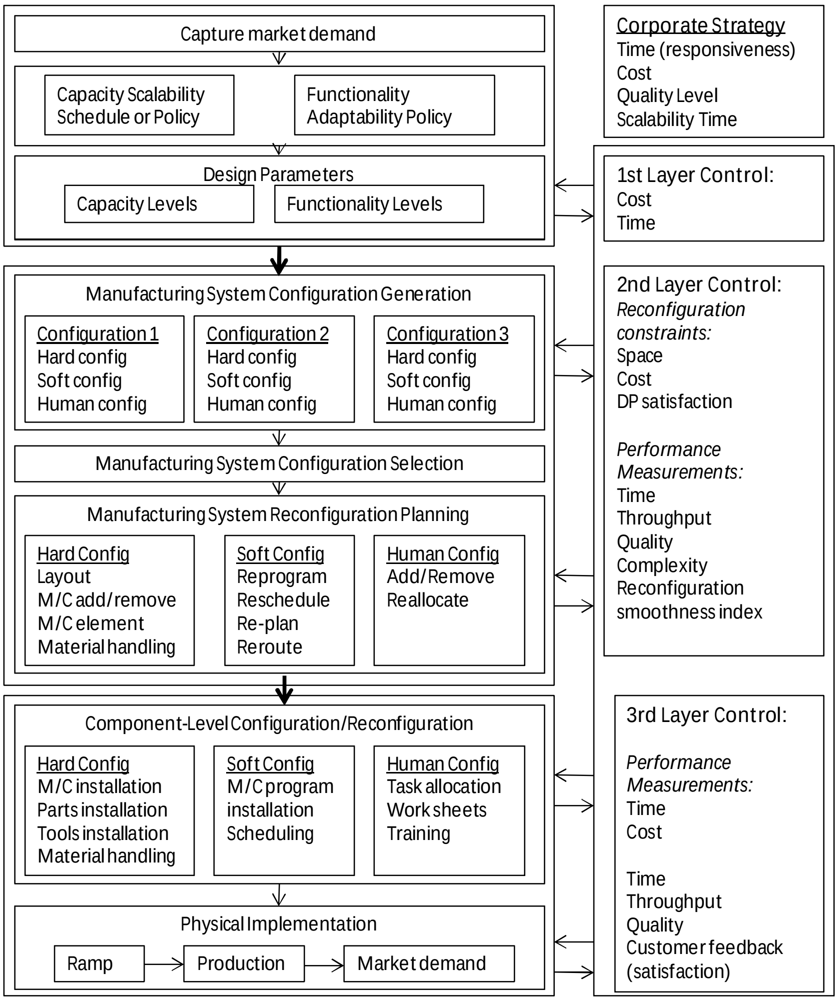

Capture market demand
Capacity Scalability Schedule or Policy
Functionality Adaptability Policy
Design Parameters
Capacity Levels
Functionality Levels
Manufacturing System Configuration Generation
Configuration 1 Hard config Soft config Human config
Configuration 2 Hard config Soft config Human config
Configuration 3 Hard config Soft config Human config
Manufacturing System Configuration Selection
Manufacturing System Reconfiguration Planning
Hard Config Layout M/C add/remove M/C element Material handling
Soft Config Reprogram Reschedule Re-plan Reroute
Human Config Add/Remove Reallocate
Component-Level Configuration/Reconfiguration
Hard Config M/C installation Parts installation Tools installation Material handling
Soft Config M/C program installation Scheduling
Human Config Task allocation Work sheets Training
Physical Implementation
Ramp
Production
Market demand
Corporate Strategy Time (responsiveness) Cost Quality Level Scalability Time
1st Layer Control: Cost Time
2nd Layer Control: Reconfiguration constraints: Space Cost DP satisfaction
Performance Measurements: Time Throughput Quality Complexity Reconfiguration smoothness index
3rd Layer Control: Performance Measurements: Time Cost
Time Throughput Quality Customer feedback (satisfaction)

Figure 9. Reconfigurable manufacturing system design architecture (Deif & ElMaraghy 2006).

Tracht & Hogreve (2011) presented two planning procedures that ease the decision-making task of a human designer during both the design phase and the reconfiguration phase of modular assembly lines. These procedures list the decision steps in a structured and logical order at a very high level. They defined three possible scenarios in which reconfiguration can occur. These are: reconfiguration due to a change in the product variant type; reconfiguration due to either an increase or decrease in demand; and, reconfiguration due to the introduction of a new product. The suggested planning phases and decision steps for reconfiguring a modular assembly system are presented in Figure 10. (Tracht & Hogreve 2011.) The architecture visualizes the different steps that need to be taken when reconfiguring a modular assembly line in response to different change stimuli. Like the method of Deif & ElMaraghy (2006) discussed earlier, although it gives an overall view of the reconfiguration design process, it doesn’t go into detail about how to accomplish the individual steps and nor is it targeted at automating any part of this planning process.

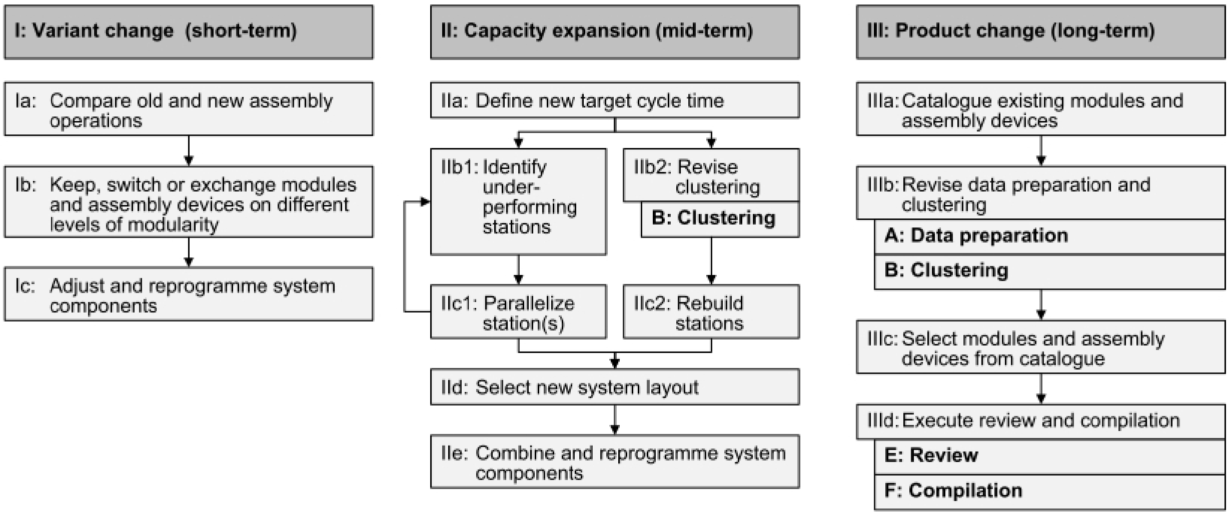

I: Variant change (short-term)
Ia: Compare old and new assembly operations
Ib: Keep, switch or exchange modules and assembly devices on different levels of modularity
Ic: Adjust and reprogramme system components
II: Capacity expansion (mid-term)
IIa: Define new target cycle time
IIb1: Identify under-performing stations
IIb2: Revise clustering
B: Clustering
IIc1: Parallelize station(s)
IIc2: Rebuild stations
IId: Select new system layout
IIe: Combine and reprogramme system components
III: Product change (long-term)
IIIa: Catalogue existing modules and assembly devices
IIIb: Revise data preparation and clustering
A: Data preparation
B: Clustering
IIIc: Select modules and assembly devices from catalogue
IIId: Execute review and compilation
E: Review
F: Compilation

Figure 10. Decision steps of the reconfiguration process for variant change, capacity expansion and product change (Tracht & Hogreve 2011).

Koren & Shpitalni (2010) presented a mathematical method for designing reconfigurable manufacturing systems. Their method takes the required manufacturing operations, their duration, and the desired production volume as inputs, and delivers the structure of the manufacturing system layout as output. The method assumes that all the machines are identical and that all the manufacturing operations required by the product can be accomplished equally well by any machine. Basically, the method delivers feasible, structural layout configurations consisting of the number of machines, the number of stages and the number of machines required for each stage based on the desired cycle time. (Koren & Shpitalni 2010.) The method does not consider the different characteristics of the machines, but assumes that the same operations take the same manufacturing time on all the machines. Neither does it consider the functional processing requirements set by the product, but assumes that all the operations can be accomplished with the given machines. Capability matching is not part of this methodology.

Reinhart and Meling (2011) presented a methodology supporting the operational planning of reconfiguration measures for automated production systems. This methodology consists of four building blocks: 1) a process model, consisting of four phases with specific goals and activities which guide the development process; 2) a model of interoperability, which attempts to describe the interoperability (i.e. interfaces) of the automated components at the technical and functional levels; 3) an evaluation model, which is used for rating the alternative concepts during the planning phase in terms of monetary and non-monetary aspects; 4) a solution library, which supports the reuse of existing coupling concepts consisting of the technical concept and the monetary and non-monetary effects. (Reinhart & Meling 2011.) The methodology provides a holistic approach to reconfiguration by integrating multiple aspects, such as coupling system components, their interfaces and monetary considerations under one methodology. Mainly it concentrates on the interfaces and the interoperation between different system components. The evaluation of the cost focuses on the costs of making the components interoperable. As with the previously discussed approaches, this approach does not make a comparison between the production task in question and the capability of the system, but simply assumes the task can be completed with the given resources, i.e. capability matching is outside of its scope.

3.3.2. Artificial Intelligence-based approaches to reconfiguration planning

According to Bi & Zhang (2001b) the reconfiguration planning of production systems is a multi-objective and complex design problem including both discrete (e.g. type of machine) and continuous (e.g. length, speed, location of the machine) variables and complex constraints. Discrete variables generate a search space with discontinuities and non-linearities. Conventional mathematical optimization techniques are hardly able to solve this type of NP hard problem in polynomial time. (Bi & Zhang, 2001b.) Therefore, methods that apply heuristics are seen as potential solutions to the reconfiguration planning problem. Artificial Intelligence (AI) -based methods have recently been widely applied to solve many kinds of combinatorial optimization and planning problems. Genetic algorithms (GA), in particular, have attracted researchers working on system design (Gen & Cheng 1997; Ho 2005; Tang 2005). According to Bi and Zhang (2001) genetic algorithms are attractive for reconfiguration planning problems, because GAs can easily handle mixed discrete/continuous variables. A few researchers have used GAs for the reconfiguration planning problem, and some of their work is discussed briefly below.

Tang (2005) treated reconfiguration planning as a “classical” planning problem, and as such it usually has three simple inputs: 1) the initial state of the world that the object resides in; 2) the goal that is desired; 3) the possible actions that can be performed. The planner’s output is a sequence of actions that, when executed in the initial state of the world, will achieve the goal. The reconfiguration planning problem can be seen as a path-planning problem for finding the shortest and cheapest path from the initial state to the desired state. Tang developed a methodology for designing and reconfiguring a Multiple Part Manufacturing System (MPMS) focusing on metal cutting. He presented a Computer Aided Reconfiguration Planning (CARP) framework using a hybrid plan-generation method based on A* algorithm and GA. The basic input to a CARP system includes the manually constructed description of the reconfigurable object, its current state, and the new performance goal to be achieved. Based upon these inputs, the CARP system will output a series of reconfiguration actions in a specified sequence in order to achieve the desired change in system performance. (Tang 2005.) The methodology concentrated on creating structural reconfigurations at the single machine level, but did not consider the whole production system consisting of multiple machines.

Ho (2005) applied motion genes to reconfigure flexible assembly line systems. He encoded and evolved conveyor components by linear and angular conveying motions. Genetic mating was used to generate alternative conveyor system layouts that satisfied specific production requirements, and the best of these was selected on the basis of ‘the survival the fittest’. (Ho 2005.) This approach only concentrated on generating feasible conveyor layouts, and ignored all the other system components.

Youssef & ElMaraghy (2007) presented an RMS Configuration Selection Approach considering the arrangement of machines, the selection of equipment and the assignment of operations. The approach consists of two phases. In the first phase, near-optimal alternative configurations for each possible demand scenario over the considered configuration periods are selected. The approach uses a constraint satisfaction procedure, real-coded Genetic Algorithms (GAs) and Tabu Search (TS) for the continuous optimization of capital cost and system availability. In the second phase, integer-coded GAs and TS are used to determine which of the alternatives produced in the first phase would optimize the smoothness of the transition within the planning horizon. A stochastic model of the smoothness of the reconfiguration across all the configuration periods within the planning horizon, according to the anticipated demand scenarios, is used for this. The criteria for system analysis are cost and availability. (Youssef & ElMaraghy 2007.)

The above-mentioned GA-based approaches require that the system is encoded into a gene and chromosome structure, and that the objective is presented as a mathematical function. Therefore, they can only provide solutions to relatively narrow problems, for which the objective function and the system encoding can be formulated with relative ease. This type of AI-based approach can't cope with the extensive and dynamic nature of the problem presented in this thesis. In this problem, the system itself, the demands placed on it, and the environment are constantly changing, so the time-consuming formulation of problems using these mathematical elements is not feasible.

3.3.3. Knowledge-based approaches to reconfiguration planning

Travaini et al. (2002) presented a knowledge-based methodology and tool for assembly line reconfiguration. Their approach is concerned with three fundamental models: a model to describe the features and functionality of the assembly line; a model to describe the features of product component assembly; and, a model that produces all the assembly sequences. An important part of the methodology is a rule-base which is supposed to compare new products facing the existing assembly lines. These models, and the rule-base, are supposed to analyse the configuration of the assembly system and the product components, and to propose and evaluate new reconfiguration solutions. As input, the methodology uses both the information collected from product component and assembly line data bases, and the information provided by automatic tools for gripper selection and sequence generation. It analyses the reconfiguration strategies, pointing out similarities and differences between the requirements for the new assembly process and the available structures of the existing assembly line. The product component data is compared with the assembly line data, and the assembly operations are allocated to the various stations on the line according to the rules. The different assembly line options are evaluated using a reconfiguration index. (Travaini et al. 2002.) This method covers the whole reconfiguration planning process from product analysis to the selection of the configuration, and also provides the information models and rules required to perform the reasoning activities. However, based on the description of the approach, it is not clear if the information structure supports automatic manipulation and reasoning with the information. The method is based on a manual comparison between the new requirements and the old assembly system with the given rules. Furthermore, it doesn't consider lifecycle information, or the interfaces of the resources.

Hirani et al. (2006) presented a methodology for the knowledge-enriched specification of the requirements for reconfiguring an assembly system. The methodology involves the elicitation of user requirements and their subsequent conversion into system requirements. The knowledge inputs from the system user and system integrator are considered. Each part of the methodology is underpinned by specialist knowledge of the products to be assembled, the assembly processes and methods needed, and the capabilities that need to be delivered for system integration and reconfiguration. The key elements in this methodology are the mapping, and its rules, which translate the user requirements into system requirements, and then into task and assembly module requirements. Once the system requirements are defined, these are compared to the specification of the existing assembly system while changes to the process elements are defined using a matching algorithm in order to derive the reconfiguration needs. (Hirani et al. 2006.)

Ratchev et al. (2007) continued this work by presenting a knowledge-based methodology for forming customizable, reconfigurable assembly cells. Their approach is based on matching user requirements to existing supplier knowledge in terms of design rules and principles, the modules offered by different vendors, new emerging technologies, and their own and their competitors' existing products. Using this approach, the decision-making involves an analysis of the requirements, the generation of assembly processing alternatives, and the evaluation and selection of the assembly modules and cells. (Ratchev et al. 2007.) Both of the above-mentioned studies are aimed at providing means and guidelines for the design and reconfiguration of an assembly system through close interaction between the customers, suppliers and the system integrators. In addition, they provide the required information in a formal, computer-interpretable format. They use matching algorithms to enable the automatic comparison of the new system requirements against the old system capabilities and the capabilities of the resources saved on the knowledge base. These methods support automatic reasoning, but their intention is not to automatize the reconfiguration planning. Furthermore, they don't deal with how to create new resource combinations to compensate for missing capability requirements during the reconfiguration planning and nor do the models deal with the resources' lifecycle information.

Von Euler-Chelpin and Kjellberg (2007) looked into the possibility of feeding the resource operation knowledge gained from the runtime environment back into the manufacturing system reconfiguration activities. Their approach didn't consider the reconfiguration process itself, but concentrated only on capturing and presenting the condition and maintenance-related information of the existing resources in order to support the reconfiguration process. They divided the capability of the manufacturing system into designed (theoretical) and actual capabilities. The actual capability depends on the restrictions arising from the process, the environment, and the wear on the resources and is measured during runtime. This shows whether the resource, in a certain context, can meet the demands and requirements posed by the product design. The context is described by the current configuration and the surrounding environment at the point when the data is captured. (von Euler-Chelpin & Kjellberg 2007.)

Harms et al. (2008) developed a framework for production system re-use, consisting of methods, tools and business process reference models. The core element of the framework is the knowledge-based approach to Computer Aided Reuse Planning (CAReP). CAReP facilitates and organizes the exchange of information between experts and information systems, enabling the logical interpretation of information. The tool is an expert system which provides the user with the knowledge utilised in the re-use planning process. CAReP aims to help the planner to decide whether to re-use a particular item of equipment based on the effort and risk involved, and company-strategic criteria. Indeed, CAReP produces system specifications and project plans for re-use, which encompass all the necessary adaptation, maintenance and modification processes, (see Figure 11). In the adaptation domain, CAReP provides rule-based solutions for assessing the interdependencies between adaptation processes and functional capabilities (e.g. the welding force of a welding gun) as well as the costs of a particular adaptation process. Using the rule-base, it's possible to generate a specification for the adaptation process for the given resource. (Harms et al. 2008.)

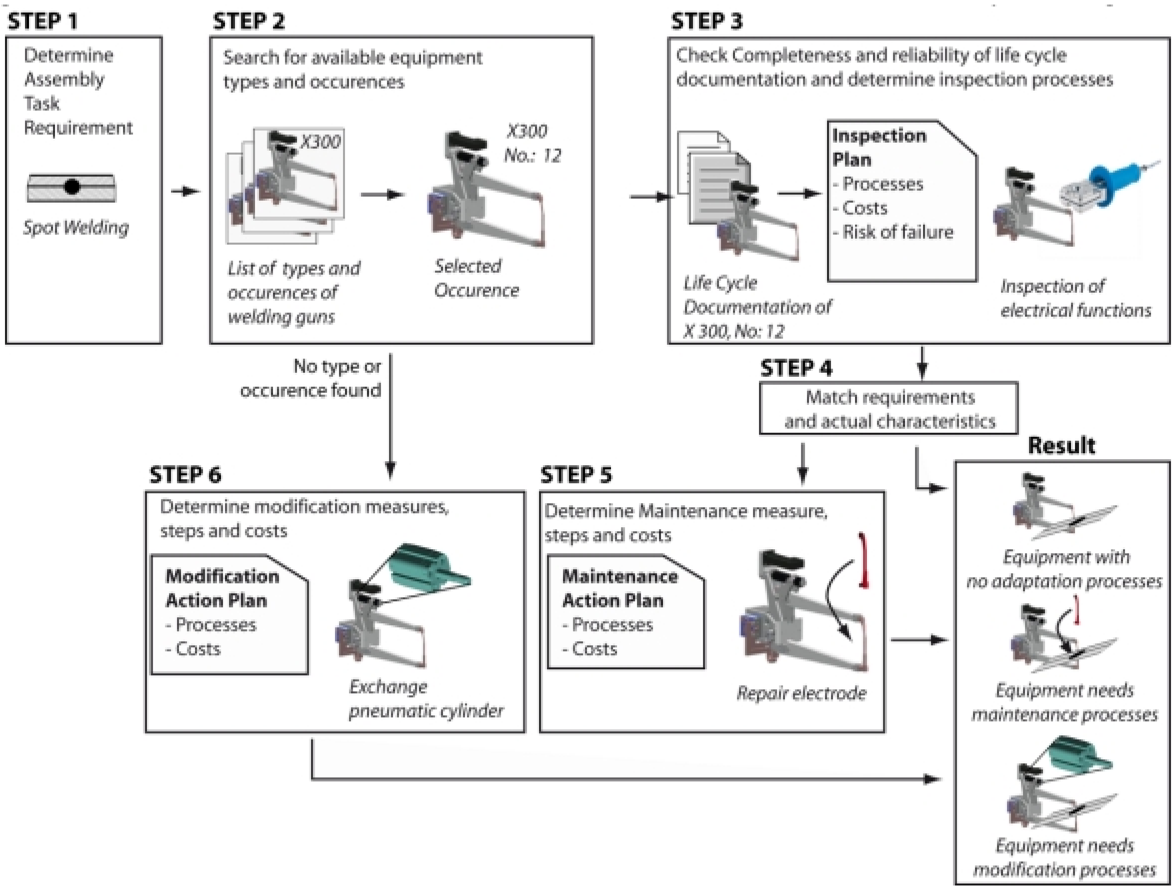

STEP 1
Determine Assembly Task Requirement
Spot Welding
STEP 2
Search for available equipment types and occurences
X300
List of types and occurences of welding guns
X300 No.: 12
Selected Occurence
STEP 3
Check Completeness and reliability of life cycle documentation and determine inspection processes
Life Cycle Documentation of X 300, No: 12
Inspection Plan
- Processes
- Costs
- Risk of failure
Inspection of electrical functions
No type or occurence found
STEP 4
Match requirements and actual characteristics
Result
STEP 6
Determine modification measures, steps and costs
Modification Action Plan
- Processes
- Costs
Exchange pneumatic cylinder
STEP 5
Determine Maintenance measure, steps and costs
Maintenance Action Plan
- Processes
- Costs
Repair electrode
Equipment with no adaptation processes
Equipment needs maintenance processes
Equipment needs modification processes

Figure 11. CAReP – Reuse options generation procedure (Harms et al. 2008).

Harms et al. (2008) use ontologies to represent the conceptual procedure and the information in a computer-based application. They extend the 'product, process, equipment' ontology of Lohse (2006) with an adaptation process and an equipment domain. They also expand the conceptual equipment definition with the failure mode concept, which includes the concepts of failure cause and effect, failure identification, and action for containment, all of which are based on a failure mode and effects analysis (FMEA). (Harms et al. 2008.) CAReP works at the device level rather than the system level. It evaluates the re-use and reconfiguration options for one device at a time and is therefore not suitable for planning the reconfiguration of complete systems. It is a rule-based expert system, where all the possible adaptation actions (including their prices) and resulting functional capabilities have to be pre-programmed into the system, which makes it feasible for only relatively static production environments. Therefore it doesn't provide a solution for the problem outlined in this thesis.

Minhas & Berger (2011) presented a concept for reconfiguring production set-ups to enable versatile production in automotive factories. Their approach starts with comparing the features of the new product with the features available in the database. The aim is to find any matches between the new case and the old cases stored in the technology data catalogue using heuristic algorithms. If a similar case is found, the process and the related parameters used for the previous case are used for the new case. The catalogue stores the application and case-specific knowledge in an ontological framework. An axiomatic optimizer utilizes axiomatic design methodology to select and optimize the design parameters of the resources based on production goals (cost, time, quality and environmental impact). The tasks are sequenced and assigned based on the ease of reconfiguration, i.e. the minimum values for time, cost and environmental impact for every change. The aim of this methodology is to enable the comprehensive re-use of existing knowledge resources and experience. (Minhas & Berger 2011.) This approach doesn't consider the capabilities of the resources during the reconfiguration design process. Instead, the selection of the resources is based solely on the previous products which have similar features. The description of this approach reveals that interface analysis is not part of the concept. Neither does it consider the lifecycle of the resources, which is regarded as an essential input for adaptation planning.

3.3.4. Capability-based approaches to adaptation

This chapter presents adaptation planning approaches which utilize descriptions of the resource capability as a basis for their reasoning. In this section, the word "skill" is used interchangeably with the word "capability" as the policy in this research is to use the original terminology for each reference when discussing them.

Timm et al. (2003; 2006) proposed an ontology-based capability management approach for multi-agent-based manufacturing systems in order to shorten the gap between process planning and production control. The task requirements and the capabilities of the resources are specified using ontologies. The approach is based on the decisions of the resource agents as to whether they can fulfil the capability requirements or not. This approach represents the capabilities based on 1st order logic. Within these predicates, the concept terms denote types of capability. The concept terms are organized in taxonomic hierarchies within the ontology, thus allowing for subsumption inference in order to obtain a more flexible and "fuzzy" match-making between the problem descriptions and the capabilities. The problem is specified using statements, which describe pre- and post-conditions. The agent's capabilities are determined by its set of available plans, which describe its ability to transform a state which supports its pre-condition into a state supporting its post-condition. They used the cobac* algorithm as a conflict measure, based on the evaluation of proximity and distance in the ontology, i.e. its taxonomic relation. STEP-NC is used to give a structured representation of the manufacturing task. (Timm & Woelk 2003; Timm et al. 2006.)

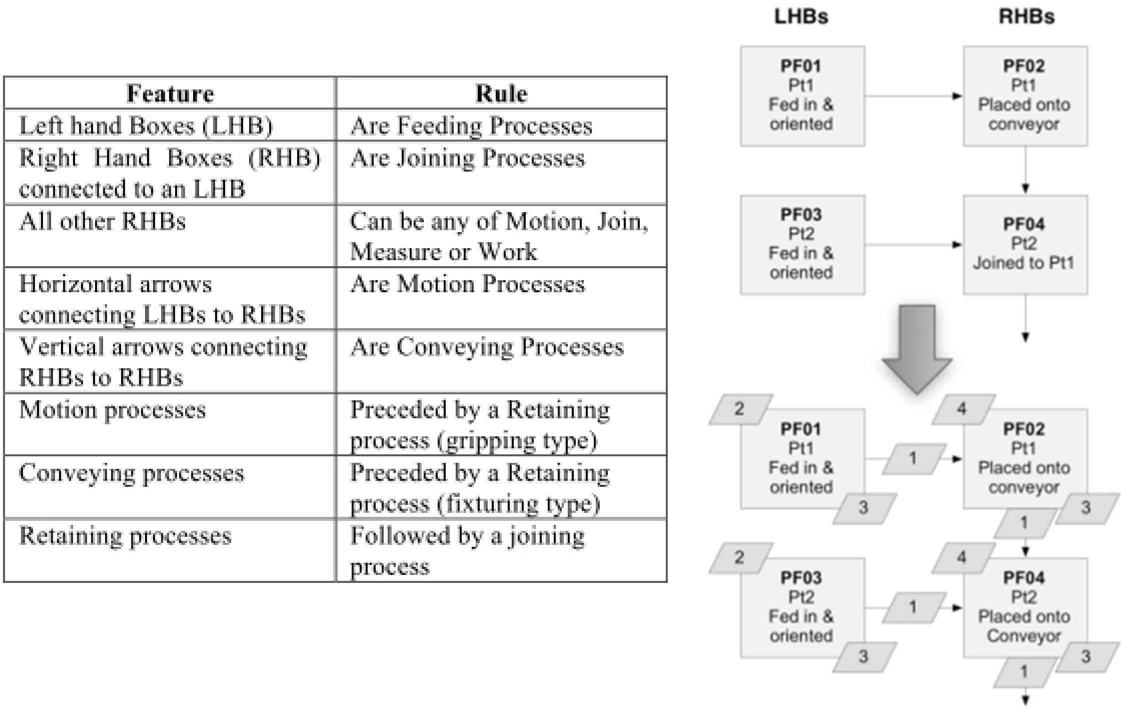
Their approach allows the process planning to occur dynamically, based on the availability of, and disturbances on, the manufacturing line and therefore it supports dynamic adaptation (e.g.re-routing or re-scheduling). Timm et al. (2006) mention combined and complex capabilities which are formed by coalitions of resource agents. However, it appears that they use those terms with different meanings than the ones used in this thesis. If all the manufacturing steps cannot be performed by one machine, then more machines will be needed to perform the sequence of required manufacturing steps to produce the product. In this approach, these machines will then form the combined capability. However, they do not specify how these combinations of capabilities are handled in the ontology. In addition, it is not specified what kind of information is included in the capability definition or whether they incorporate any parameters. The creation of new combinations of devices, including matching their interfaces, is also neglected in this approach.

In the SIARAS project (Skill-based Inspection and Assembly for Reconfigurable Automation Systems) an intelligent system, called the skill server, was built to support automatic and semi-automatic reconfiguration of production processes. As the basis for its reasoning, the skill server needs both the available knowledge about the production technology (i.e. generic knowledge of the system and its skills and possible tasks), and the workpiece to be produced and its related tasks. The ontology expressed in OWL is used to model this knowledge. The skills are modelled as a taxonomy following standards like DIN 8580 and VDI 2860. Each device has a set of skills which are regarded as instances of the taxonomy skills. Atomic skills are defined as skills which cannot be broken down into any further activities. For example, gripping would not be an atomic skill because it includes lower-level operations like, 'move until the proximity sensor is active', 'enable the valve until the touch sensor has been enabled' and so on. Atomic skills can be directly linked to the tasks. The ontology is used to reason whether skills match particular tasks or whether devices have those skills. However, the skill server's reasoning is not based on the ontological representation of the knowledge alone, as specific knowledge of the domain is also needed. This knowledge is put into utility functions, which are plugged into the skill server core. The utility functions can be either device-specific (e.g. for setting up a certain sensor) or application specific (e.g. path planning for an industrial robot) or they are meant to solve any other kind of problem. (Bengel 2007.)

According to Malec et al (2007) the reconfiguration approach of SIARAS is based on the user modifying the current process description in some way (parameter change, task exchange). After validating and accepting the user input, the skill server analyses the new process regarding its feasibility (do the devices provide all the skills required for the updated task), accuracy, robustness, reliability, timing, power consumption, collision avoidance, cost, and so on. In the final phase of reconfiguration, they use the mechanism of ontology-based compilation in order to create configurations, and to facilitate a smooth transition from a generic process representation to a device-specific code. (Malec et al. 2007.)

The SIARAS approach does not meet the objective of this thesis because it uses the skill descriptions to control production and programme the system. Therefore, the skills have been defined on a detailed action level. With regard to adaptation, their approach doesn't take the lifecycle of the resources into account in the skill description. Another drawback is that the skill definitions don't include the parametric properties of the skills, but the properties are assigned to the devices rather than to the skills. The name of the property instance is used to indicate the parameter value of the property, such as "Electrical interface_connectorM12-4Pins" (SIARAS 2008). This makes it difficult to compare similar skills in different types of devices. In the SIARAS approach the devices are classified in a hierarchical class structure. These are then associated with the relevant skills and properties, e.g. devices belonging to the gripper category will have gripper skills and devices belonging to the sensor category will inherit the sensor functions. However, this may not be the most optimal way to assign the skills, because it is not certain that all the same types of devices really have the same skills. Also, multifunctional devices often have more skills than are assigned to the device category they most closely fit into.

Smale and Ratchev (2009) proposed a capability-based approach for multiple assembly system reconfiguration. Their work comprises a reconfiguration methodology supported by a capability model and a capability taxonomy. The capability model combines the roles of the requirements definition, the capability definition and the capability comparison and delivers aggregated capability sets. The capability taxonomy is suited to both equipment specification and requirement definition and its main purpose is to allow the capabilities to be defined with terminology that is common to both equipment suppliers and product manufacturers. The capability taxonomy has been built based on six capability classes, which are representative of the majority of processes associated with manufacturing and assembly, namely: motion; join; retain; measure; feed; and work. The product requirements are converted into required capabilities through the use of a Process Flow Template and the Capability Taxonomy within the Capability Identification Process. The Process Flow Template is based upon the principle that assembly requires two parts to be fetched and maintained together. Therefore, the core assembly processes are “Moving Part x” and “Joining Parts x and y”. Using this principle as the basis for the definition of the required capabilities, a number of rules can be applied. Figure 12 shows the rules and the capability identification process. (Smale & Ratchev 2009.)



Feature	Rule
Left hand Boxes (LHB)	Are Feeding Processes
Right Hand Boxes (RHB) connected to an LHB	Are Joining Processes
All other RHBs	Can be any of Motion, Join, Measure or Work
Horizontal arrows connecting LHBs to RHBs	Are Motion Processes
Vertical arrows connecting RHBs to RHBs	Are Conveying Processes
Motion processes	Preceded by a Retaining process (gripping type)
Conveying processes	Preceded by a Retaining process (fixturing type)
Retaining processes	Followed by a joining process

LHBs
RHBs
PF01 Pt1 Fed in & oriented
PF02 Pt1 Placed onto conveyor
PF03 Pt2 Fed in & oriented
PF04 Pt2 Joined to Pt1
2
PF01 Pt1 Fed in & oriented
3
1
4
PF02 Pt1 Placed onto conveyor
3
1
2
PF03 Pt2 Fed in & oriented
3
1
4
PF04 Pt2 Placed onto Conveyor
3
1

Figure 12. The rules applied to the Capability Flow Diagram and an example of the Capability Identification Process (Smale & Ratchev 2009).

After making the Capability Flow Diagram for the product, the Capability Taxonomy and associated guidelines are applied to define each individual capability. The output is a numerical value which is then logged within the relevant set. The final step is to perform a comparison of the existing and required capability sets with a Comparison Matrix. (Smale & Ratchev 2009.) The reconfiguration methodology presented in (Smale & Ratchev 2010) is based on a known set of products. Finding the proper production sequence, i.e the order in which products are produced, is an integral part of the methodology. The production sequence is achieved through adopting a product-centered approach and identifying the commonalities between each of the products. The configuration analysis is first performed for each product independently, at which stage the capabilities to remove, retain, investigate and procure are identified. This is done by comparing the existing capabilities with the required ones using the capability matrix. After this, the sequence analysis is performed. (Smale & Ratchev 2010.)

This methodology (Smale & Ratchev 2009; Smale & Ratchev 2010) supports reconfiguration planning by providing a means to compare the product requirement with the existing resource capabilities, using the capability taxonomy and rules. However, although it is intended to computerize the methodology in the future, at present the definition and matching of capabilities is done manually, and therefore requires a lot of effort. There are a few reasons why this approach doesn’t support the objective of this thesis. Firstly, the available system components are treated as individual items and their interfaces, co-operation and layout are not taken into consideration. In addition, this approach does not consider the capability parameters, the combined capabilities of multiple devices or the lifecycle information and current status of the devices. The approach can be extended from the device level to the unit and station levels, but it doesn't examine how to derive the unit or station level capability. In other words, it relies on a human expert to make the definition and comparison of the capabilities. Also, because the reconfiguration methodology is based on a known set of products, and optimization of the production sequence based on minimum downtime and changes, it can't handle an unpredictable environment in which the future product requirements are unknown.

3.3.5. Agent-based and holonic approaches to adaptation

Most of the agent-based and holonic approaches presented in this chapter also utilize some sort of capability description and are therefore of special interest here. The reason for not including these approaches in the previous chapter is that their focus is more on controlling the production, rather than planning the adaptation. Therefore they differ slightly from the approaches presented in the previous chapter.

Actor-based Assembly Systems (ABAS) mechatronic reference architecture, developed by Martinez Lastra (2004) in his dissertation, aims to meet the requirements of flexible, adaptive and reconfigurable assembly systems. In ABAS, the reconfigurable systems are built by autonomous mechatronic devices, called actors, which deploy auction- and negotiation-based multi-agent control in order to collaborate towards a common goal, i.e. the accomplishment of assembly tasks. These assembly tasks are complex functions, which are generated from a combination of simpler activities called assembly operations, which are the individual goals of the actors. The actors' functionality is mapped to these atomic assembly operations. Each of these actors provides one specific service (operation) to the society. (Martinez Lastra 2004; Martinez Lastra et al. 2009.) The approach is targeted purely at agent-based shop floor control. From the viewpoint of this thesis, it is interesting because it deals with atomic assembly operations which subsequently form higher level assembly processes once the actors have been combined into clusters. However, as the description of the operations (skills) doesn't define the parameters of these skills, this approach is not suitable for detailed resource selection.

Al-Safi and Vyatkin (2007) proposed an ontology-based reconfiguration agent that attempts to reconfigure the manufacturing system without human intervention after detecting changes in either the requirements or the manufacturing environment. It creates a new reconfiguration by analyzing the new requirements and using the ontological knowledge model to infer facts about the environment, and is thus able to deduce whether the current environment can handle the given manufacturing requirements or not. The reconfiguration agent consists of four main components, namely, a requirement analyser, a floor analyser, a knowledge modeller and a decision engine. The decision engine decides whether the requested operations exist in the current manufacturing system, and whether they are connected with logistic operations, and then proposes new configuration based on this. If the requirements are not satisfied, it lists the missing operations. (Al-Safi & Vyatkin, 2007.) This approach does support automatic adaptation in that it compares the new requirements against the existing system. However, it only takes into account those machines which are currently in use. Although it checks whether the current environment is able to perform the new required operations, it doesn't try to search for the missing technology from resource libraries, or to create new combinations of the existing machines. The reasoning between the requirements and resources is based on pre-defined operations which it is known that the resource can perform, and therefore the approach doesn't consider the properties and capabilities of the resources in sufficient detail.

Barata et al. (2008) presented a multi-agent-based control architecture for a shop floor system (CoBaSa) which supports fast re-engineering and plug and play capabilities. In their approach, each manufacturing component is agentified in order to enhance its adaptability and interaction competences so that it can respond to different requests. These agents, each of which represents one manufacturing component, can be aggregated through the Broker Agent GUI to form a coalition of agents. This coalition coordinates higher-level processes (complex skills) based on the skills available among its members. The Broker Agent GUI works as an interface between the agent system and the human expert, through which the expert can choose which components and associated skills they want to aggregate in order to provide a possible new complex skill. (Barata 2006; Barata et al. 2008.) According to Cândido and Barata (2007), the ontology is used to identify which basic skills are necessary to provide complex skills. After gathering the coalition members' skills, the Coalition Leader Agent searches the ontology for any complex skills that could possibly be supported with the basic skills. The Pallet Agent contains the information about the order in which the skills need to be executed in order to assemble the product. (Cândido & Barata 2007.)

For this thesis, the CoBaSa is interesting because it deals with the skills and complex skills of multiple co-operating agents. Unfortunately, it has not been reported in detail how the skill information is handled in CoBaSa, such as how it deals with the skill parameters in the ontology. Based on the example presented in Barata et al. (2008) they use the skill concept name to indicate the properties of the skill (e.g. Drill_Small_Skill, Drill_Large_Skill, Drill_Special_Skill). None of the reasoning seems to be based on the technical properties of the devices as the skills don't actually have any parameters; just the concept name. The CoBaSa architecture is intended to provide agility at shop floor control level. Therefore, it is targeted more at operational control than at planning an adaptation.

Frei (2010) applied the CoBaSa in Self-Organizing Evolvable Assembly Systems (SO-EAS). However, because the coalitions of agents in the original CoBaSa architecture was static, and formed manually by the user, she developed mechanisms for dynamic coalitions in order to overcome this limitation. A chemical reaction model was utilized for this. (Frei 2010.) SO-EAS are systems which are capable of self-organization while the assembly system is being created, and self-management during the production time. They are composed of modules with local intelligence and self-knowledge, able to self-organize to form suitable dynamic coalitions and shop-floor layouts, which fulfil the generic assembly plan provided by the user. This is done based on the rules for self-organization. During production, the production modules self-manage themselves while executing the assembly tasks based on the policies for self-management. (Frei et al. 2010.) Several rules are applied for the formation of the dynamic coalitions based on the chemical reaction model. These include rules for interface compatibility, composition pattern, the creation of composite skills, task coalition matching, layout creation, transport linking and transforming the generic assembly plan into a layout-specific assembly plan. (Frei 2010; Frei & Di Marzo Serugendo 2011.)

As stated by Frei et al. (2009), ontologies and the complementary XML representation of the resources is used to guide the creation of composite skills from simple ones. The XML-file attached to each resource agent specifies what other capabilities are needed with that specific resource. Also, as discussed with the CoBaSa approach, the skill definitions don't include parameter values. Instead, each skill with different parameters is saved separately in the resource information with the skill name indicating the skill parameter, e.g. Drill20 and Drill30 for drilling holes of 20 or 30 mm. This kind of approach to handling the skills inhibits the reusability of the skills between different resource types, in addition to which it means that there is a huge amount of information which has to be handled when dealing with resources having numerous skills.

Ferreira et al. (2010) presented a multi-agent architecture for the requirement-driven reconfiguration of modular assembly systems. Their approach was driven by a set of requirements which addressed the specific needs of precision, modular assembly systems, and catered for both the physical and logical constraints of the modules as well as their joint emergent behaviour. The agent architecture consists of a Requirement agent, an Equipment Module agent, a Physical agent, an Assembly Process agent, an Assembly Process broker and an Equipment broker. This architecture aims to assess the effort required for reconfiguration, as well as to provide solutions from the existing modules when the product or process requirements change. (Ferreira et al., 2010.) The strength of the proposed method is that it is able to consider different manufacturing capabilities capable of fulfilling the same requirement, and to assess them according to the cost of the reconfiguration. However, the paper does not specify how much of this reasoning can be automatized and how much is left for human expertise. Neither does it specify how the information about the agents is presented.

Zäh et al. (2010; 2011) presented a holistic approach to cognitive job control of production systems. Their approach is based on so-called smart products, which store the knowledge about the production process and its current state. The cognition is implemented by RFID tags attached to the products. Their multi-agent-based approach concentrates on product-based production planning and shop floor control, and enables the plans to be adapted dynamically. The product-specific operation times are measured and stored on the smart product during the runtime. This product-specific data can be used for learning and updating the overall planning data, and improving future plans. (Zäh et al. 2010; Zäh et al. 2011.) Zäh et al. (2010) use a holistic data model, consisting of product-, process- and resource-related data, represented in an XML structure. In order to present both the process requirements and the resource capabilities in terms of feasible operations, they use the DIN8580 standard and VDI2860 guidelines. In their approach, the production sequences are presented in a graph-based model. The capability profiles of the resources represent the executable production of every single resource. These are used to allocate the production steps to the resources. (Zäh et al. 2010.) According to Zäh et al. (2011) the neutral building-block-based description of working processes facilitates the flexible configuration of the work plan, the consideration of potential alternative sequences in order processing, the easy modification of single building blocks and the integration of additional working processes.

The capability profile used in the cognitive factory (Zäh et al. 2010; Zäh et al. 2011) directly indicates the operations the machine is able to execute, i.e. the product manufacturing steps have a clear correspondence with certain resources. This approach does not involve reasoning about the capabilities, or matching them against the product requirements. This kind of approach is efficient for applications where the set of possible products is fixed, but their production volumes and orders vary. This enables the possible product-operation-resource combinations to be predetermined, and the planning and control can thus concentrate on routing the product according to resource availability and economic parameters. However, it is not suitable for adaptation planning and reactive adaptation in an unpredictable environment.

3.3.6. Conclusions about the existing approaches to adaptation

Chapters 3.3.1 to 3.3.5 summarised the different approaches taken by a number of researchers to support the adaptation of production systems. The strengths and weaknesses of each approach for solving the research problem of this thesis were briefly discussed. The review doesn't claim to be complete, and it has to be recognised that there may be other approaches which have not been discussed above. However, the review was extensive enough to show that there are a wide range of different approaches to adaptation planning and the reactive adaptation of production systems. Despite the significant amount of research carried out in this area, none of the existing approaches are in themselves able to solve the problem of this thesis.

Youssef and ElMaraghy (2007) presented a literature review of the various approaches to systems configuration selection. They identified some gaps in the existing approaches, which also directly relate to the adaptation problem. Firstly, most of the work done treats the configuration problem from one particular perspective, i.e. the physical arrangement of the machines, or it deals with configuration as a parameter without defining it. What is lacking is the automatic generation of feasible alternative configurations for different demand scenarios. Secondly, most of the previous work only focuses on the cost and economic benefits in order to evaluate performance, but it doesn't consider other system evaluation criteria. (Youssef & ElMaraghy 2007.) Table 2 summarises the main limitations of the existing approaches to adaptation in terms of this thesis problem.

Table 2. Limitations of the existing adaptation approaches.

Limited coverage	
	The approaches usually consider only a certain small part of the overall problem. They may, for example, concentrate on finding the best possible structural configuration at the single machine level, or on finding the best layout to optimize the throughput or to minimize the costs. In other words they don't support all the aspects of adaptation that are seen as relevant in this work. These include the ability to support physical, logical and parametric adaptation, the consideration of the old system layout and its components, the new product requirements, the capabilities of the system components, the lifecycle information of the system and its components, and lastly, the possibility of incorporating different user-defined criteria into the reasoning process.
Lack of suitable resource identification and selection	
	Most of the approaches don't consider the matching of product requirements with resources having suitable capabilities. The suitable resource options are either pre-defined or the features that can be manufactured with the machines are already described, eliminating the reasoning. The emphasis is then on the optimization of the configuration based on some user-defined criteria, rather than finding the right resources for the given requirement. Simply put, the suitable resource identification is beyond the scope of most approaches.
Lack of suitable resource models	
	The previous point highlights the lack of suitable resource models, which would allow the matching of product requirements against resource capabilities. Most of those approaches that do utilize some sort of capability description don't incorporate the capability parameters into the description or are not able to deal with the combined capabilities of multiple resources.
Lack of automatic generation of feasible configurations	

	Automatic generation of feasible alternative configurations for different demand scenarios is missing. The approaches reviewed don't deal with finding or generating new resource combinations for the new requirements. This, again, is due to the lack of suitable models for describing the resources and their capabilities.
Static resource descriptions lacking the lifecycle aspect	
	Those approaches which do utilize resource descriptions as a basis for adaptation deal with static resource descriptions. This means that the capabilities of the resources are not updated dynamically based on the gathered lifecycle information. The lifecycle or resource condition information is therefore not used as a basis for decisions about re-use or adaptation. This is a serious problem, because the resource behaviour is often strongly dependent on the resource lifecycle and usage history.
Lack of interface considerations	
	Most of the reviewed approaches don't put any emphasis on the interfaces and compatibility of the resources while making the configuration decisions.
Inability to evaluate the magnitude of needed change	
	Very few approaches try to evaluate the magnitude of the change the production system needs to undergo when the product requirements change. This would be important in order to be able to estimate the effort and cost of the required adaptation. Those approaches which do estimate the costs are based on pre-defined information of the price of certain adaptation actions. These "expert systems" cannot deal with unpredicted situations and are therefore very limited in their scope.

In a rapidly changing production environment the creation of a new production system configuration has to be based on rapid match-making between the requirements and offerings. If neither the requirements nor the offerings are pre-known, it is not possible to pre-define their direct relationships. Nevertheless, this was the procedure followed in most of the reviewed approaches, which thus eliminated match-making between the requirements and the offerings. Therefore, such approaches can't solve the problem of the adaptation of systems in a constantly changing environment. Nevertheless, a few approaches did base the adaptation on this match-making, this being enabled by the descriptions of the resource capabilities. Using a process-oriented definition of capabilities, it is possible to make a match between the process requirements and the resource capabilities (i.e. requirements and offerings). The ability to describe the capabilities of resources is seen as an important enabler for successful adaptation. Therefore, it is emphasised strongly in this work.

3.4. Information models supporting adaptation

As discussed above, resource capability modelling forms an essential part of this thesis and therefore attention has been paid to reviewing the existing approaches to describing resources. The aim of bringing automation to the design and adaptation of a production system requires a structured knowledge representation of the resource properties and its constraints. The manufacturing resource information and the capability descriptions are regarded as the basis for various manufacturing activities, including process planning, resource allocation, system and facility design, controlling the operations and planning the adaptation of a system. If these descriptions are formal and expressive enough, they enable automatic computerized methods to find suitable system components, and to test alternative scenarios in the early phases of the design.

As stated by Vichare et al. (2009), the standardised representation of information in the manufacturing business has traditionally focused on the products themselves, with information relating to their design, geometry and the processes required for their manufacture. What are lacking are standardized information models which represent the manufacturing equipment used to produce such products. Resource data models and tools for different system components exist, but they are either vendor-specific or very limited in their scope. This is an important omission, because that is the information which is required when the production systems are actually built. (Vichare et al. 2009.)

Recently, manufacturing resource modelling has been attempted by several researchers using different methods, having different purposes, from different viewpoints and at different levels of detail. The representation of machine tools, in particular, has been widely studied; see, for example, (Hedlind et al. 2010a; Hedlind et al. 2010b; Kjellberg et al. 2009b). These representations are often based on the ISO 10303 standard, also known as STEP. Tolio et al. (2010) listed some standards that are used to describe machine tools and cutting tools, such as ISO 13399, entitled ''Cutting tool data representation and exchange'', ISO 10303 – IAR 105 on the kinematic modelling of manufacturing resources, and the standard ASME B5.59, which defines information models and formats for describing machine tools for milling and turning based on XML data format. However, as stated by Tolio et al. (2010) available standards dealing with manufacturing resources often do not consider their integration in the plant, neglecting the understanding of the production system as a whole. As the goal of this thesis is to provide a more generic resource description, which is able to describe a multitude of different types of resources, those approaches aimed purely at machine tool representation are not discussed in any further detail here.

The following section presents a few interesting reference solutions for describing resources and their capabilities, and for matching the product requirements with suitable resources. First, ontological approaches to describing resources are discussed, followed by the resource description languages and then object-oriented approaches. The section concludes with a summary of the limitations of the existing approaches to resource description.

3.4.1. Ontological approaches to describing resources

Ontologies play an important role in knowledge-based modelling. They provide a standardized way to present knowledge from different domains and from heterogeneous knowledge sources. Ontologies are developed to support the exchange of meaningful information across autonomous entities that can organize and use the information heterogeneously. According to Gruber (1993), "an ontology is a formal, explicit specification of a shared conceptualization". The conceptualization is applied to a limited domain, such as the product, process and system domains. The conceptualization aims to break down the different terms and entities of this domain into well-defined and distinctive concepts. The concepts are expressed in a formal way in order to allow computers to use them. The concepts have to be explicit in order to avoid inconsistencies and ambiguities in meaning. This can be achieved by using non-ambiguous classifications, relations or metrics. Lastly, the conceptualization has to be shared and agreed upon by the different user groups in order to provide a common means of communication and frame of reference. (Gruber 1993.) Below, a few approaches which utilize ontologies to describe resources and their related domains will be introduced.

Core Ontology – an ontology representing product – process – system information

Research done in the FP6 IP-PiSA project by Lanz et al. (2008) and Lanz (2010) resulted in the Core Ontology, common knowledge representation (KR) and semantics, which connect the product, process and system domains under one reference model. The main goal in this development was to include the meaning in the model and to enable different design tools to interoperate across the design domains. The structure of the representation was formed according to the requirements set by the knowledge management and integration challenges of different design tools. (Lanz 2010; Lanz et al. 2008.)

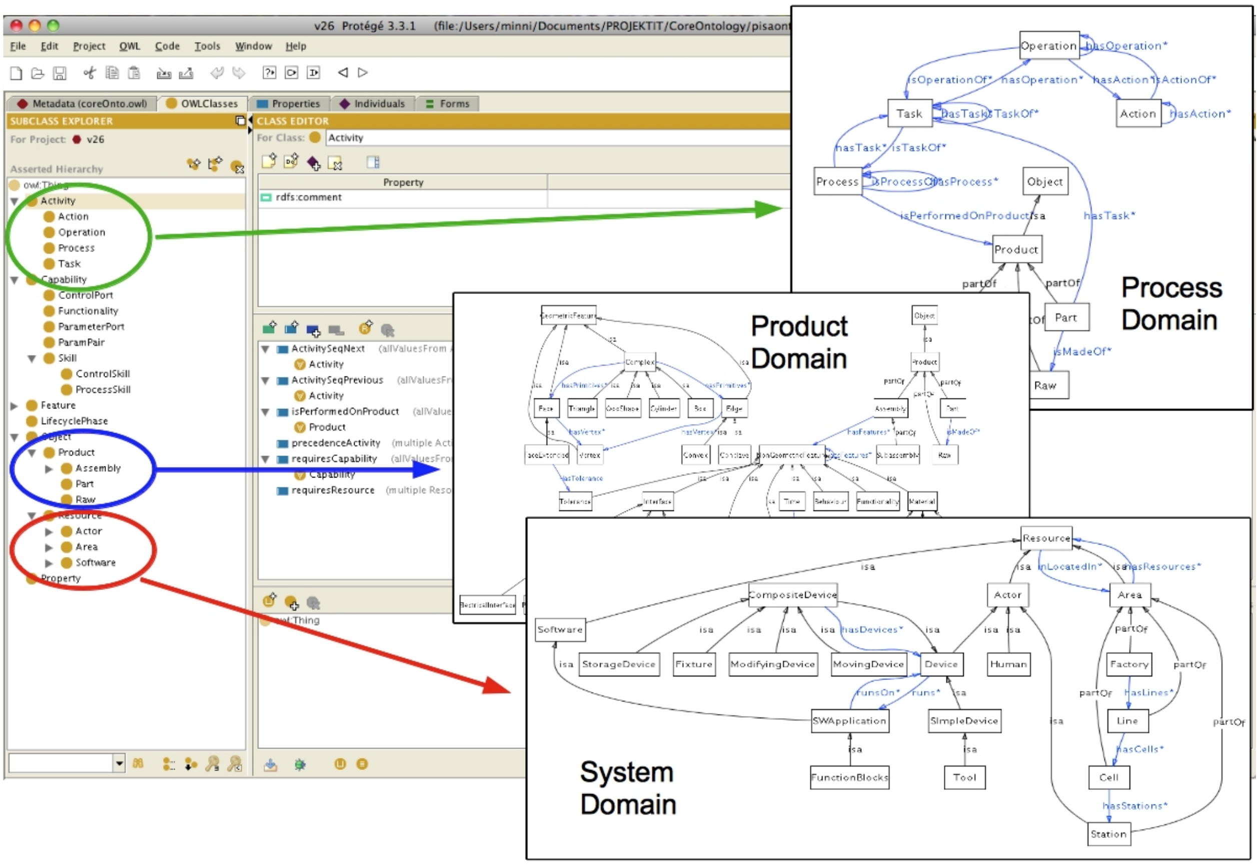


Figure 13. Core Ontology (Lanz 2010).

The Core Ontology, presented in Figure 13, formalizes the representation of knowledge between the product, process, and system domains utilizing fractal systems theory as a guiding principle. The product model consists of classes for products, assemblies and parts, going into the level of geometrical features, such as primitives (e.g. box, cylinder, cone), edges, faces and face extendeds. The non-geometric features, such as dimensions, coordinates, material and surface information, are also included in the product domain. The process domain has a straightforward approach to representing the process levels. The system domain was created to represent the production facility and resource information. This model allows resource connections to parts, assemblies and products via the activity class. The ontology is saved into a Knowledge Base (KB) which is described in detail in (Lanz 2010.) The original version of the Core Ontology doesn't provide any means for describing the capabilities, lifecycle information or interfaces of the resources. So, during the course of this thesis project, it has been extended to better suit the requirements of production system adaptation planning and reactive adaptation. These extensions are discussed in Chapter 4, and more specifically in Chapter 4.5.

Function-behaviour-structure framework

Several approaches have been reported, (Lohse 2006; Kitamura et al. 2006; inter alia), for the ontological definition of a device's or module's capabilities based on the function-behaviour-structure (FBS) framework. The FBS framework was originally developed by Gero (1990) and is illustrated in Figure 14, below. The main focus in FBS is on understanding the functional capability of devices based on their behaviour and structure.

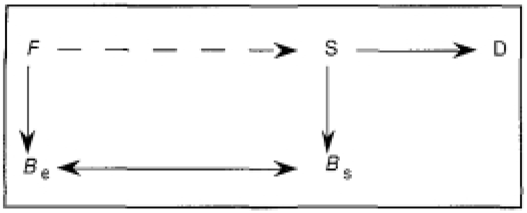


Figure 2. Model of Design as a Process.

B_e = Set of expected behaviors
B_S = Set of actual behaviors D = Design description
F = Set of functions S = Structure
→ = Transformation
---> = Occasional transformation
<–> = Comparison

Figure 14. The function-behaviour-structure framework (Gero 1990).

Kitamura et al. (2006; 2010) developed an ontological framework for modelling the functional knowledge of devices. They stated that functional knowledge reflects part of the designer's intention (design rationale). How the function is recognized depends on the system, the environment, the situation or the use, as opposed to objective data, such as shape and structure. The distinction between function and behavior, used by Kitamura et al., originated from research into qualitative reasoning. The term "behavior", when applied to a device, represents temporal changes of the physical quantities of some physical entity (operand) other than the device. Behavior is objective and independent of the context (which includes the designer's intention), the user's aims and the system in which the entity is embedded. The term "function", on the other hand, is related to the intention of a designer or a user, and is hence context-dependent. A behavior can perform different functions depending on the context. Moreover, a function can be performed (realized) by different behaviors. Simply put, function answers the question "what to achieve", whereas behavior describes "how to achieve". (Kitamura et al. 2006; Kitamura & Mizoguchi 2010.)

Kitamura et al. (2006) discussed functions as a role. By role they meant the concept that an entity plays in a specific context. A function can be defined as a role played by behavior in a teleological context. The teleological context for an engineering system, in turn, can be determined according to the designer's or user's intention. (Kitamura et al. 2006.) The difference between Kitamura's function-behavior approach and the traditional function-behavior-structure paradigm as applied by, for example, Lohse et al. (2006b) is that Kitamura's behavior does not describe the internal behavior of the resource arising from its structure. Kitamura et al. don't go to the structure level when describing resources.

Separating "what to achieve?" from "how to achieve?" is seen as beneficial to adaptation planning, and particularly for dynamic reactive adaptation. If these two can be separated, the product requirement can just state "what?" leaving more freedom for the resource selection based on the resource's availability, or other criteria.

A Matchmaking Methodology for Supply Chain Deployment in Distributed Manufacturing Environments

The work performed at NIST (National Institute of Standards and Technology) by Ameri and Dutta (2008) aims to connect the buyers and sellers of manufacturing services in web-based e-commerce environments. The matching is based on their semantic similarities in terms of manufacturing capabilities. The proposed matchmaking algorithm operates in Manufacturing Service Description Language (MSDL), which is an ontology for the formal representation of manufacturing services. The work uses a graph-matching approach, since the MSDL descriptions can be represented as directed, labelled trees. The fundamental concepts underlying MSDL are supplier, service and resource. The relationships between these concepts can be seen in Figure 15a. Manufacturing services are an important part of the methodology and are formally defined using the MfgService class. Any instance of the MfgService class has at least one manufacturing process, and is typically characterized by an array of manufacturing capabilities. An MfgService is enabled by certain manufacturing resources, i.e., machines, tools, fixtures, etc. Figure 15b presents an example of subclasses of the MfgCapability class for describing the machining capabilities of a service. (Ameri & Dutta 2008.)

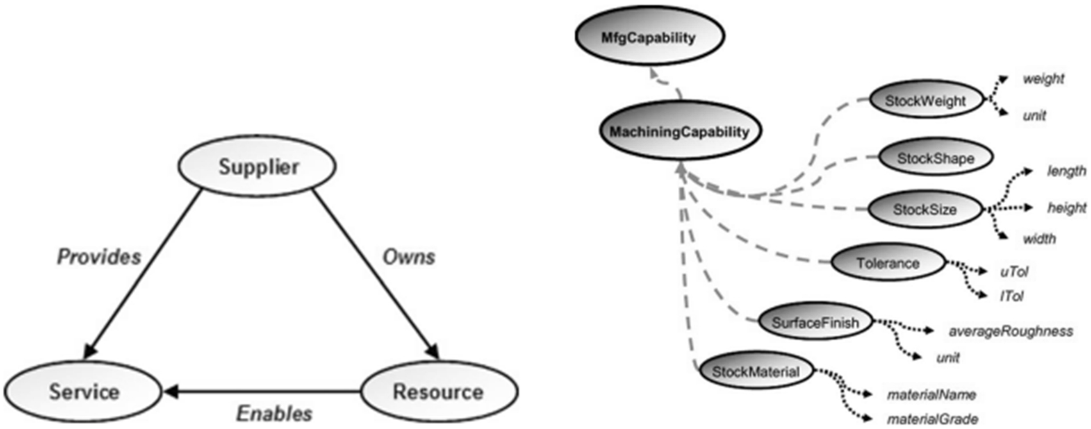


Figure 15. a) Relationships between supplier, service and resource; b) Example of MfgCapability subclasses (Ameri & Dutta 2008).

MSDL enables a uniform treatment of knowledge from suppliers and demanders by modelling them both as generic concepts which need to be matched. Supply and demand are instances of advertisement and query, respectively, which are both represented through service and supplier classes. Accordingly, machine agents can use their internal reasoning mechanisms to determine the semantic similarity between the instances of advertisement and query, based on their formal descriptions and taxonomies. (Ameri & Dutta 2008.) The underlying problem to be solved in this approach is somewhat similar to the problem presented in this thesis. Supplier corresponds to the resource, and demander to the product requirement. The approach is based on matching the existing capabilities with the required ones, but again, MSDL doesn't consider how to combine the capabilities of multiple co-operating resources. Actually, it only describes the combined capabilities, because it describes the capabilities of a supplier as a whole, but fails to specify each individual resource which is involved in the manufacturing operations.

EUPASS – Equipment ontology and skill concept

During the EUPASS-project, briefly discussed above in Chapter 3.1.4, the EUPASS ontology for modelling evolvable, modular, ultra-precision assembly systems was developed. The ontology was divided into three sub-domains, namely Product, Equipment and Assembling Process. In the EUPASS approach, the Skill concept is used to bridge the gap between processes and equipment in the ontology. The skills are divided into basic skills and complex skills. The basic skills are the most fundamental skills, whereas the complex skills are combinations of various simple skills. (Lohse et al. 2008.)

Lohse et al. (2006a) presented an ontological equipment model which was intended to support the effective design of reconfigurable assembly systems. The ontology follows the function-behaviour-structure framework and includes a separate formalism for the specification of a module's capability and interfacing requirements. The equipment ontology is used to represent the functionality and behavior of the equipment arising from its structure. For example, the functionality of a robot is described by dividing the robot structure into links and joints, and describing the behavior of each link and joint separately. The connectivity between different equipment components is handled through interface ports. Figure 16 presents an overview of the equipment ontology concept and the relation between function concepts and equipment concepts. (Lohse et al. 2006a.)

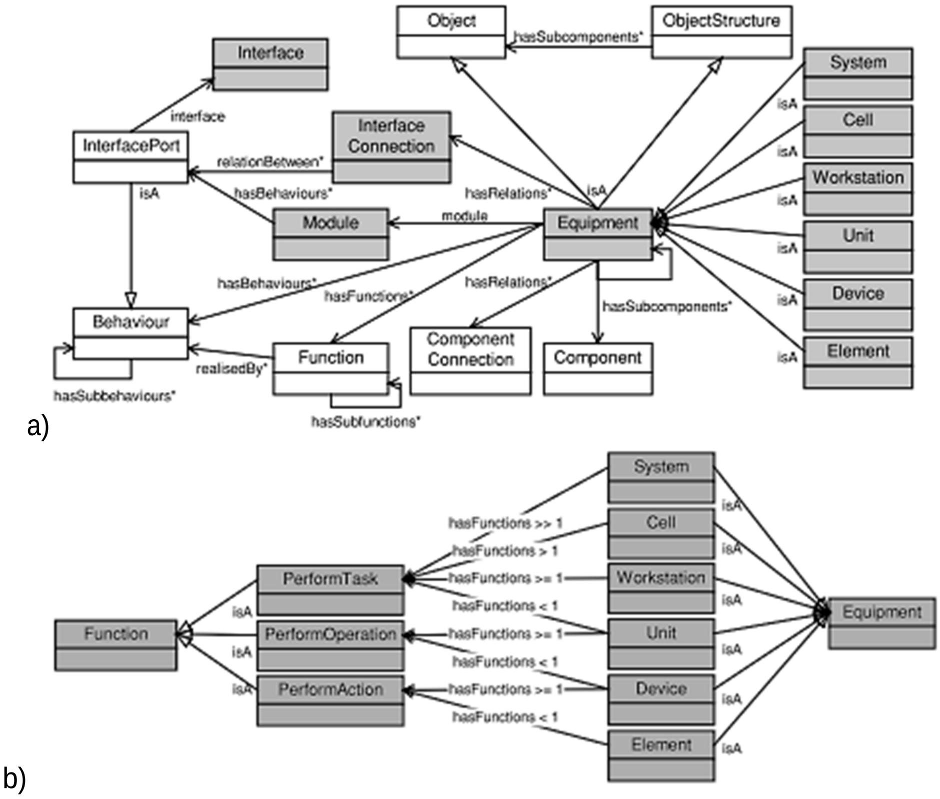


Figure 16. a) Equipment module ontology concept overview, b) Relation between function concepts and equipment concepts (Lohse et al. 2006a).

Lohse et al. (2006b) divide the assembly processes into three hierarchical levels, namely task, operation and action. They use rule-based activity decomposition patterns to recognize the overall function from the set of lower-level functions. The activity decomposition patterns refer to the methodology by which the assembly process is specified through stepped increases in the level of detail from tasks, through operations up to actions. The process decomposition patterns define the required sub-activities for a specific type of activity and include their temporal and logical constraints. This definition results in an AND/OR graph-like structure that links higher-level activity types to lower-level ones. (Lohse et al. 2006b.)

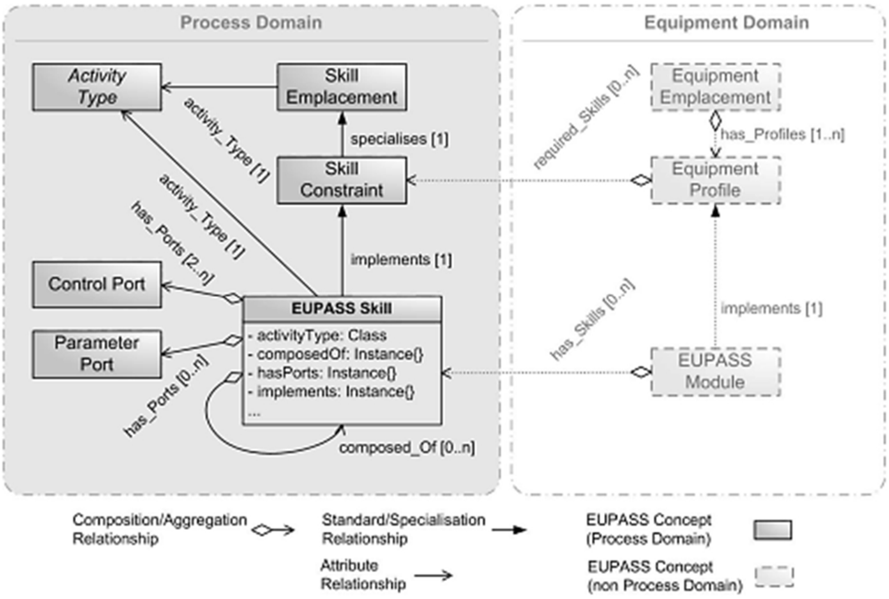


Figure 17. EUPASS Skill Definition Overview (Lohse et al. 2008).

Figure 17 shows an overview of the EUPASS Skill concept and its relationships to other associated concepts within the process domain. The boundary conditions, or rather the way a skill can interact with its environment, are defined through Control and Parameter ports. They both have input and output ports and variables, which are identified and described by the Emplacement and Blue Print discussed in the next chapter, but their value changes during the operation of the module. The Control Ports take part in the control flow, or in enabling the skill, and they govern the sequential behavior between skills. Each skill has to have at least one input or 'begin' control port and one output or 'finished' control port. The sequential control behavior can be defined by mapping the input and output control ports to each other. The Parameter Ports govern the exchange of parameters between skills. For example, if the aligning skill needs to position something based on a position determined by a measurement skill, then the measurement skill needs to have a position parameter output port and the aligning skill needs to have a position parameter input port which can be mapped to each other. (Lohse et al. 2008.)

The EUPASS approach recognizes the problem that individual items of equipment are not usually able to execute the required task alone, and therefore combinations of devices providing complementary skills are needed. For example, in order to move items a robot, which provides the moving skill, needs to be combined with a gripper, which provides the holding skill. Unfortunately, the EUPASS documentation doesn't provide details on how their approach deals with the emergence of the atomic skills into complex ones, or how they present the complex skills within the ontology model. As far as one can tell, the skill information doesn't include parameters, and therefore the problem of combined capabilities is still unsolved.

3.4.2. Resource description languages

The EUPASS Approach – The Emplacement and Blue Print concept

The Concept of the Digital Manufacturing Module Description, i.e. the Emplacement concept, was developed by Siltala et al. (2008) to give a standardized description of the EUPASS modules which comprise an assembly system. The aim of these descriptions is to aid in both the design and commissioning of assembly systems and the design of each individual module, and they are intended to be utilized by anyone dealing with the assembly system (e.g. the module providers, the system integrators or the end users). The key terms in the Emplacement concept are Emplacement, Profile and Blue Print. They aim to provide a digital representation of the production modules and include essential information like mechatronics, interfaces, skills, technical and business properties, CAD models, datasheets and so on. XML (eXtensible Markup Language) is used as a format for the Emplacement and Blue Print languages. (Siltala et al. 2008.)

Emplacement is a generic description of the features and general requirements of a module for an assembly system. It is a collection of interface and property specifications for particular modules. Examples of Emplacements are a gripper, a manipulator and a feeder. Each Emplacement contains one or more Profiles. The Profile is a technical, schematic and detailed description of a module. It defines the detailed set of features and requirements that makes one module interconnected and interchangeable with another. A Profile is a representation of one kind of physical module and it can be instanced by physical hardware. Examples of profiles are a vacuum gripper, a 2-finger force-feedback gripper and a 3-DOF Cartesian manipulator. Blue Print (file) is an electronic specification of an actual, existing process module. The Blue Print file content needs to correspond to the Profile and Emplacement specification, of which the Blue Print is an instance. In addition to these, there is also a History Container file. While the Blue Print file is the same for all instances of the module, the History Container contains relevant information about the specific instance of the module, and is updated during the lifecycle of the module. (Siltala et al. 2008; Siltala et al. 2009; Siltala & Tuokko 2010.)

The Emplacement concept can be used as an information source to support various planning and control activities, including planning the adaptation of production systems. The concept includes the history information and the skills of the resources, which play a crucial role in enabling the planning for re-use and adaptation. The incorporation of the interface definitions in the model supports the integration of resources in different configurations. However, again, modelling the combined skills is not part of the concept.

Unified Manufacturing Resource Model (UMRM)

Vichare et al. (2009) developed a Unified Manufacturing Resource Model (UMRM) to represent manufacturing equipment, in particular CNC machining systems. It aims to capture information related to the manufacturing facility and its capabilities, and to provide support for the automation of decision-making in process planning. The model represents the machine resource and also its additional auxiliary devices, such as the workpiece and cutting tool changing mechanisms, bar feeders, fixtures, conveyors, pallet/gantry systems and robotic arms. UMRM is STEP-NC compliant and it follows the EXPRESS-G data modelling methodology. (Vichare et al. 2009.)

UMRM is based on modelling kinematic chains of machines. With UMRM, any machine tool or auxiliary device in the CNC machine can be considered as an assembly of various mechanical machine elements with different kinematic joints to facilitate constrained movements, and thus can perform the desired manufacturing operations. The UMRM's entity, the *mechanical_machine_element*, is an abstract representation of any mechanical machine element in the CNC machining system. The *kinematic_joint* is used to represent possible kinematic links between two machine elements. The kinematic properties of the movements are represented with the entity *axis*. When several axes are attached to a mechanical machine element, this represents a machine element capable of moving in different directions in order to perform the intended manufacturing operations. For example, a robotic arm assembly can be modelled in this manner to represent both its movement and the workspace configuration. These entities are presented in Figure 18. (Vichare et al. 2009.)

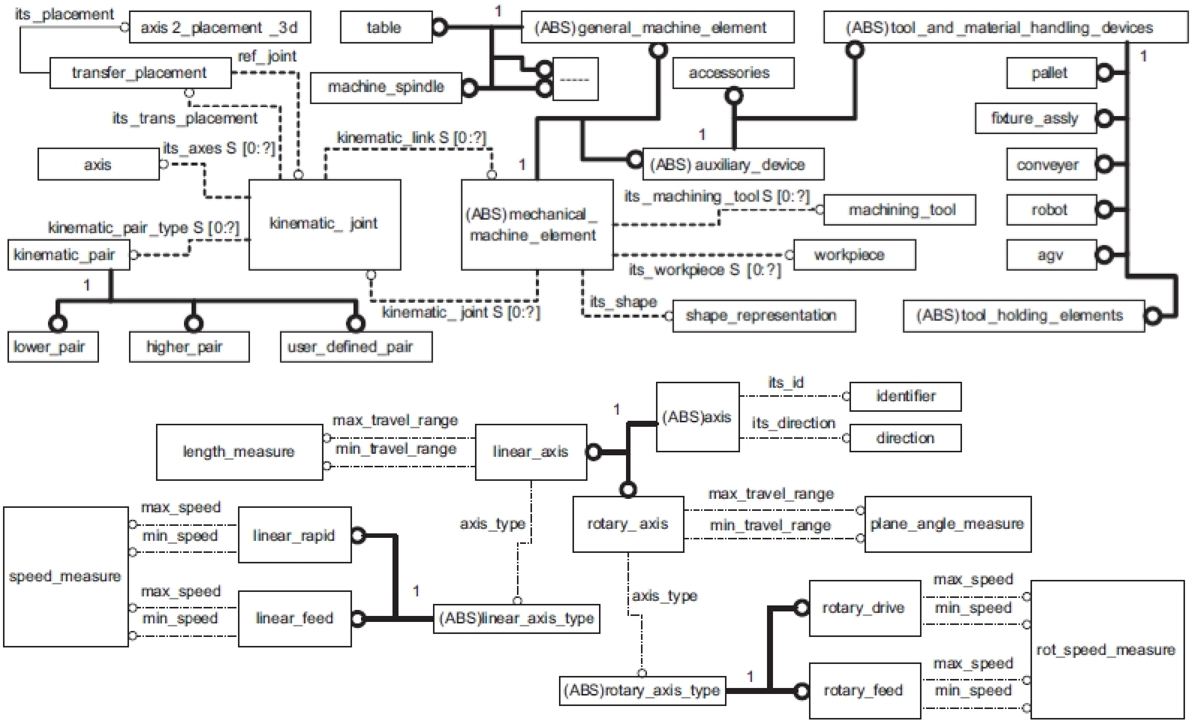


Figure 18. UMRM entities mechanical_machine_element, kinematic_joint and axis (Vichare et al. 2009).

The UMRM is based on modelling the kinematic chains and therefore, in this regard, it works at a very detailed level. The UMRM is mainly meant for modelling the CNC machine resources, whose process capability depends strongly on the different axes attached to the machine element. Compared to, for example, an assembly system composed of multiple individual devices (without the devices being divided into elements like joints and axes), this kind of modelling is not adequate. The UMRM is not able to capture the combined capability of multiple co-operating resources. It is a static model, which describes the behavior of one resource at time. Therefore, it doesn't provide a feasible solution to the resource description problem presented in this thesis.

3.4.3. Object-oriented approaches to resource description

Maropoulos et al. (2003) presented an object-oriented resource model which was developed to support dynamic, aggregate planning of manufacturing operations within production networks during the early stages of product design. Aggregate planning aims to measure a product's manufacturability and to evaluate alternative design configurations and manufacturing scenarios through the allocation of multiple parts to remote facilities within the supply network. This resource model is based on a library of two sets of classes: resources, meaning the physical entities; and resource types, indicating the process-related capability of the resource. These resource types (e.g. handling, transport, machining) are associated to the individual resources. For all the resources, several properties are given, such as location, set-up times for parts and batches, cost rates, utilization, quality, etc. Each resource type has a different set of operating parameters, ranging from the maximum depth of cut to the tool speeds and feed rates. These resource keys are exactly the same as for processes, enabling direct mapping between processes and resources (see example in Figure 19). (Maropoulos et al. 2003.) Central to the aggregate process model classes are the algorithms, which estimate the Quality, Cost and Delivery performance based on the feature characteristics of the product model and the operation parameters of the resource, in order to facilitate resource selection for a given product (Maropoulos et al. 2002).

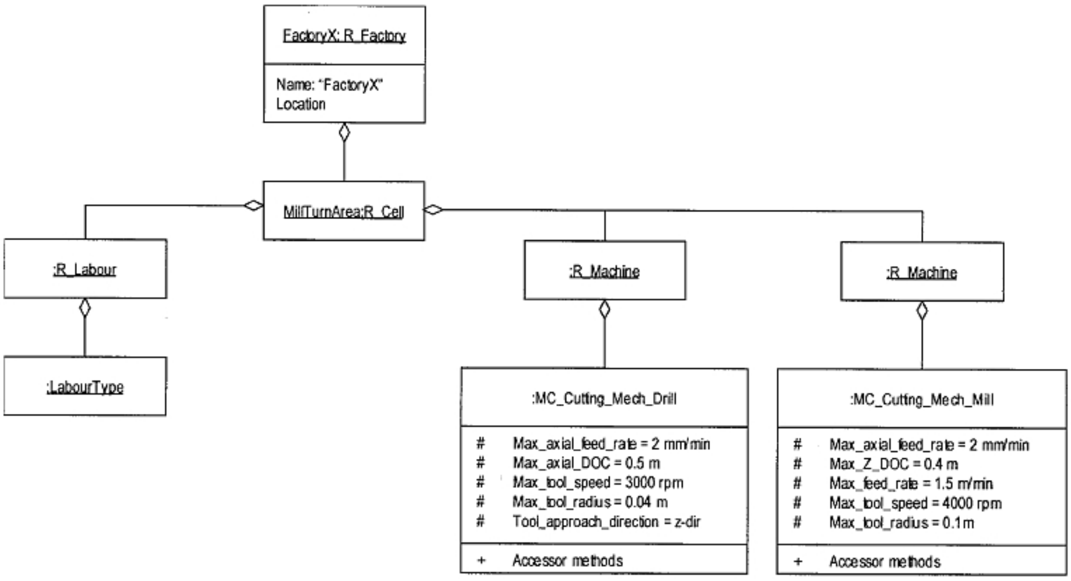


Figure 19. Example of resource model (Maropoulos et al. 2003).

Even though the model presented by Maropoulos et al. (2003) is able to describe the process-related capabilities of resources, it has a few drawbacks when applied to the problem in this thesis. It is based on the traditional classification of resources. In other words, the resources are divided into types based on their assumed common functionality. It doesn't support modelling the combined capabilities of resources and it doesn't include the lifecycle information or interface descriptions in the model.

Colledani et al. (2009) presented a conceptual framework for the integrated modelling of product, process and production system data. The aim of their work was to provide a common data structure as a reference for different methodologies and tools used during manufacturing system design. The framework was developed as an object-oriented model using UML (Unified Modelling Language) and translated into a relational database. The data framework was enriched with the evolutionary concept, which allows for the consideration of possible changes to the system (i.e. reconfigurations) as well as to the products and processes. The product information was presented in the classes of Workpiece, Machining Feature, Scenario Node and Production Problem. The data related to the manufacturing system were detailed by the classes of System, Selected System, Hyperplane, Machine, Carrier, Load/unload Station, Physical Pallet, Tool, Tool Carrier and Performance

Evaluation. The process was described with the classes of Machining Operation, Machining Workingstep, Workplan, Pallet, Setup Face and Setup WP. Several attributes were given for each of these classes. The dynamics of the manufacturing system configuration are represented by the "previous system" attribute, which links a given system with its previous configuration. (Colledani et al. 2009.)

The above-mentioned approach by Colledani et al. (2009) gives a very comprehensive and straightforward way to describe the properties of the resources, as in the catalogues and the selected process parameters. The approach is not meant to describe the capabilities or combined capabilities of resources. The relational database approach gives a concrete way to save the information generated during the system design process in a structured way, but it doesn't provide a basis for reasoning in order to extend the knowledge. While the ontologies allow new information to be added to the model relatively easily, the databases are very stiff and the use cases (requirements) need to be known before implementing the system. Ontologies allow more flexibility to extend the model. Therefore the relational database approach is not feasible for the problem presented in this thesis.

3.4.4. Conclusions about the existing approaches to resource description

The preceding chapters introduced and discussed some of the existing approaches to resource description. At the same time, their contributions and limitations in terms of supporting the resource description problem in this thesis were briefly discussed. Although not exhaustive, the review did highlight some aspects of the information models that need to be developed further in order to better support production systems in changing environments and to allow for the automatic generation of different configuration scenarios.

Usually, different classification systems are used to describe production systems. Traditionally, the manufacturing processes have been divided into subtractive, additive and transformative processes and the resources have been classified into groups based either on their common properties or the functions (processes) they perform (e.g. milling machine, lathe, robot, conveyor, etc.). This division has traditionally been good enough to classify the machinery. However, the emerging new hybrid technologies of today combine many types of processes into one device. Therefore, describing resources solely by the traditional means of classification has many limitations. If the resource is only allowed to be a member of one class, then a multifunctional device needs to be forced into the most appropriate class, even though it might have functionalities which belong to a number of different classes. One example of this kind of machine is the multifunctional universal CNC lathe, which is able to perform turning, milling and drilling. Also, depending on the context, the same device may be used to perform different activities (take a different role), e.g. the same device may be used to screw or drill, depending on the tool attached to it. Describing the resources through classification doesn't support the easy formulation of functional combinations either, as it must be based on pre-defined rules for matching resource types. Therefore, the traditional classification of resources is not expressive enough for the problem at hand.

Many of the approaches reviewed here have tried to overcome the limitation discussed above, in which the resource's functionalities are assumed to be purely based on the resource type, by incorporating the capability or skill definition into the resource description. Based on the process-oriented capability definition, it is possible to make a match between the process requirements and the resource capabilities. However, with regard to the current thesis problem, there are two main limitations to the existing information models. These are summarized in Table 3.

Table 3. Limitations of the existing resource description approaches.

Lack of ability to model combined capabilities	
	Even though the problem of combined capabilities (or complex skills) was mentioned in a few of the reviewed approaches, no solution to the problem was defined in detail. These approaches defined the logical relationships between the simple and complex skills, but didn't incorporate the parameter information into the capability definition. Therefore, they didn't consider how to combine the parameters of the simple skills in order to derive the parameters of the complex skills. Therefore, it appears that the problem of describing combined capabilities of multiple co-operating resources still remains unsolved.
Lack of lifecycle information of resources	
	Most of the reviewed approaches didn't include lifecycle information of the resources in the resource description. They were mainly static resource models, describing the nominal capability of the resources rather than their actual, current capability. While the resource is being used, its condition and capability may vary and therefore it would be beneficial to take this lifecycle information into account in the resource description. With regard to the adaptation problem, it is crucial to know the current capability of the resource in order to be able to more accurately estimate, e.g. its reusability or its ability to cope with the new requirements.

Based on the review, ontologies seem to be a feasible way of representing knowledge for describing the knowledge elements needed when dealing with the problem of production system adaptation. Ontologies allow new information to be added to the model relatively freely, and this eliminates the necessity of knowing all the use cases before constructing the ontology. Compared with, for example, relational databases, ontologies allow more flexibility for extending the model and adding new use cases when needed.

4. PROPOSED METHODOLOGY FOR ADAPTATION

This chapter presents the computer-aided adaptation methodology which was developed for this research and introduces the solutions to each of the defined sub-objectives. Figure 20 reiterates the problem definition and supplements this with the solution elements, which will be discussed in this chapter.

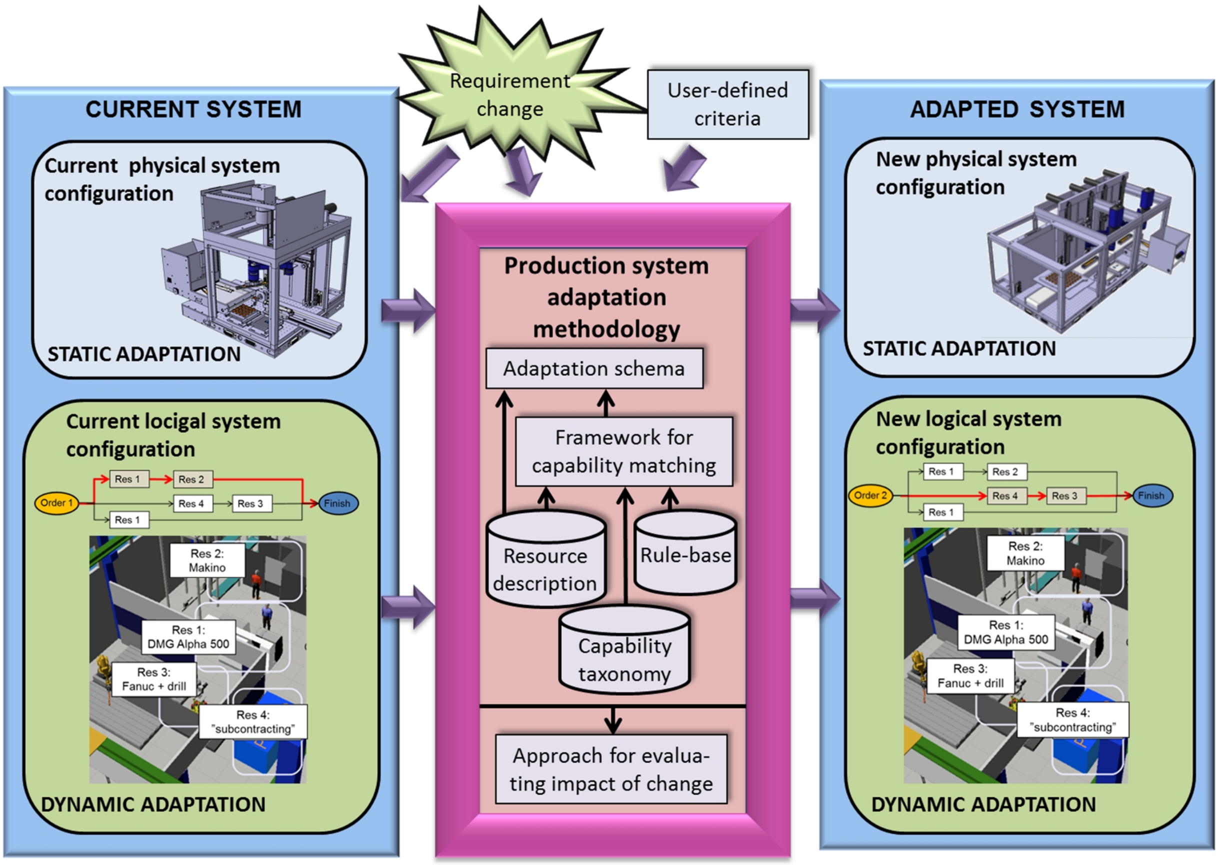


Figure 20. Problem definition and solution overview.

The core of this adaptation methodology lies in the capability-based matching of product requirements and system capabilities in the context of the adaptation process. The computerized matching of available resources against product requirements requires a formalized and structured representation of the functional capabilities, properties and constraints of the resources. A formal information model that is able to describe the existing system, as well as to formalize the elements that can be added to or removed from the adapted system is required. A description of the products and processes is also required as input for the adaptation planning. The last requirement is, of course, a schema to guide the adaptation process and indicate the required activities and information flows that are needed during the adaptation planning and for dynamic reactive adaptation.

The requirements for the information models are introduced in Chapter 4.1. After that, the capability-based resource description and the components of the overall resource description are introduced in Chapters 4.2 and 4.3 respectively. Chapter 4.4 presents the framework and rules for the capability-based matching of product requirements and system capabilities. Chapter 4.5 introduces the actual implementation of the formalized resource and capability model. The adaptation schema, which forms the backbone for the adaptation methodology and guides the use of the other elements and information resources in the adaptation process, is introduced in Chapter 4.6. Finally, Chapter 4.7 discusses an approach to evaluating the impact of product requirement changes on the production system. This approach is based on the other results obtained during this work and presented in the earlier chapters.

4.1. Definition of requirements for the information models

The following elements – product (order), process and resource – need to be formally modelled in order to support the adaptation of production systems based on the automatic matching of product requirements against the production system capabilities. In practice, detailed product and process modelling are beyond the scope of this thesis and have been tackled by another researcher (Garcia et al. 2011). However, as they are crucial input data for the capability matching, this work has set some requirements for these, which will also be described below. Table 4 below lists the requirements for these three information models.

Table 4. Elements to be formally modeled and their requirements.

Element	Requirements and characteristics
Product & Order	The product and order model needs to capture the product design, quality requirements and production volume. If a single part is to be manufactured, the product model needs to describe the workpiece in terms of its geometrical dimensions and features (e.g. face, pocket, hole, slot, etc.) that need to be accomplished. The features define the technological requirements for the resources. If a product is to be assembled, the product model needs to present the parts, their relations and liaisons, and the assembly features.
Process (capability requirement)	The process model should define the manufacturing and assembly processes so that it can capture all the relevant requirements needed to define suitable production systems and devices. The resource selection needs to consider both technical and temporal constraints. Technical constraints are directly derived from the features and liaisons that need to be established, i.e. from the product model. Temporal constraints are defined by the order in which the activities need to be performed. Therefore, the process model should describe the required activities and their temporal and logical order relationships. The processes need to be modelled from the product perspective, i.e. what needs to be done to the product in order it to transform into a finished product (e.g. grasp product, move product). The resource perspective (e.g. open fingers, close fingers), even though important from the production control point of view, is not required here. The required activities should be formally expressed in terms that are closely related to the formalism used to describe the available resource capabilities. The process model should enable the modelling of processes at different abstraction levels (function vs. behaviour) depending on the situation.
Resource	The resource model needs to be able to describe the characteristics of single resources as well as complete production systems. The resource description has to support at least the following three activities: 1) selection of suitable system components, i.e. devices; 2) combination of the selected

	components into suitable production system solutions; 3) evaluation of alternative system configurations. The focus in the resource model has to be on describing the functional capabilities of the resources. In order to enable effective selection of suitable resources for a given requirement, the resource model also needs to include the parameter values by which the resource performs the functions. The capabilities need to be defined so that they can be synthesized into higher level capabilities (combined capabilities) when two or more resources are combined into one functional configuration. The capability model should also enable the integration of the resources into those combinations with the required functionality. There needs to be a definable relationship between the process definition and the resource capability specification. The resource model should contain a clear definition of the connectivity constraints between individual resources. In other words, the interface description, which supports decisions about the compatibility of two resources, needs to be included in the resource model. The system description needs to describe the location of the equipment in the resulting configuration in order to enable spatial reasoning about the existing resource functionalities. The resource model is intended to be used in the context of the adaptation process. This means that the resources involved in the process are not brand new, but each is in its own lifecycle state. Therefore, the model must take the lifecycle of the resources into account. The behaviour of the resources is constantly changing and thus the representation of the resources cannot be static. It needs to be updateable in order to stay valid and fit for its purpose.

For the problem at hand – the production system adaptation planning problem – the actuality is an important trait for the models. According to Avgoustinov (2007), the actuality refers to the time-dependent accuracy of the model, meaning that if the modelled object changes over time, its model should be updated in order to remain relevant and fulfil its purpose. The actuality may change in two situations. On the one hand, it can “expire” if the object being modelled is changed or new information about it becomes available. On the other hand, new scientific discoveries can also make a model outdated and require its actualization. (Avgoustinov 2007.)

In changing environments like a production environment, it is important to work with up-to-date information in order to be able to make informed decisions. In this thesis, the resource model actuality is of supreme interest. During their individual lifecycles, resources are described by many different models. Some of the models can be considered static, like the resource structure and designed nominal capability. Models that describe how individual resources are used, for example factory layout, can also be static, but only for a limited period of time. Models describing the behaviour or performance of individual resources are dynamic due to wear, maintenance, repair or other changes to the resources, and thus need to be updated during their lifetime. These aspects need to be considered while developing the resource description.

4.2. Resource description based on capabilities

Each device in the production environment has certain properties and behaviors. Some of these properties and behaviors allow the device to perform certain processes. The properties of the devices have certain ranges and the functionality of the devices is restricted by certain constraints. These can be, for example, environmental constraints like the maximum permitted humidity and temperature of the operating environment, or technical properties, such as the maximum torque of a spindle or the velocity range of a moving axis. Automatic matching of available devices against product requirements requires formalized and structured representations of the functional properties and constraints of the devices.

As was pointed out earlier, most of the approaches to resource description tend to classify the resources into groups based on their common properties, or the functions they provide (e.g. milling machines, lathes, robots and so on). Unfortunately this kind of classification limits the expressiveness of the representation. To overcome this limitation, instead of classifying devices, in this thesis the functional capabilities of the devices are classified. This way, one device may have multiple capabilities which can be used in different contexts. Furthermore, new capabilities may be discovered during the resource lifecycle and these can be assigned to the device when, and if, they emerge. For the capability classification, the capability taxonomy has to be constructed so that it allows reasoning between different levels of abstraction.

There are multiple definitions for the word “capability” in the literature, and often the word “skill” is used instead. Holmström (2006) defined a manufacturing capability as follows, “A manufacturing system’s capability is the inherent ability of the manufacturing system to perform change in the properties of a material in order to produce tangible products.” With regard to this thesis there are two shortcomings in this definition. Firstly, it doesn’t include assembly systems. Secondly, it doesn’t consider those capabilities which don’t change the properties of the material, but which exist in the production system and are definitely needed in order to be able to change the properties of the material. For example, visual sensing itself doesn’t change the properties of the material, but it may be needed in order to be able to pick up the material and assemble it into a product. Therefore the definition of capability is extended to:

Capability is the inherent ability of a resource or system to perform change in the properties of materials, parts or assemblies, or to perform activities which may be required to change the properties of materials, parts or assemblies, in order to produce tangible products.

It needs to be noted that intangible services also require capabilities. However, because this thesis concentrates on discrete production systems, these are excluded from the definition.

In order to help the reader with the remainder of the text, Table 5 summarises the capability-related concepts which are used in the following chapters. A more detailed explanation of each term is given in the relevant chapters.

Table 5. Definitions of capability-related terms.

Term	Definition
Capability	Capability is the inherent ability of a resource or system to perform change in the properties of materials, parts or assemblies, or to perform activities which may be required to change the properties of material, parts or assemblies, in order to produce tangible products. A capability is a functionality of a resource, such as "milling", "drilling", "screwing" and so on.
Simple & combined capability	Capabilities are divided into simple and combined capabilities. Combined capabilities are upper level capabilities, which can be divided by functional decomposition into simple, lower level capabilities. Combined capabilities are combinations of other (simple or combined) capabilities.
Capability concept name	Capabilities are characterized by their name and their parameters. The capability concept name indicates the natural name of the capability, such as "moving".
Capability parameter	Capability parameters describe the characteristics of a capability, e.g. the "moving" capability is characterized by *"velocity"* and *"acceleration"* parameters. They help to distinguish between different resources which have similar capabilities.
Strong capability	Strong capabilities are capabilities that fit into the original definition of capability. They are functionalities of resources.
Weak capability	Weak capabilities don't fit into the original definition of capability, because they don't provide any functionality. They are properties of the resources, which don't naturally relate to any other simple capabilities, for instance, "basicDeviceInfo", which describes the size and weight of the resource.
Capability model	A capability model is used for modelling and managing the combined capabilities formed by combinations of simple capabilities. In the model, the simple and combined capabilities are linked by capability associations. The instantiated capability model consists of the pool of capabilities that may exist in the production system and their parameters.
Capability association	Capability associations are links between simple and combined capabilities. There are two types of capability associations, inputs and outputs. The simple capabilities provide output associations while the combined capabilities require input associations.
Capability assignment	Capability assignment is used in the ontology to assign resource-specific capabilities to the resources. Capability assignment contains the resource-specific parameter values for the capabilities.
Capability taxonomy	The capability taxonomy categorizes the capabilities in a hierarchical structure. The taxonomy is used to enable mapping between product requirements and resource capabilities at different levels of detail and allow subsumption-based reasoning about the capabilities.

The resource description approach defined in this thesis is based on ontological modelling. The Core Ontology defined by Lanz (2010) and introduced in Chapter 3.4.1 is used as a basis for describing the product, process and system-related information. It allows the basic descriptions relating to the resources to be formalized. In the scope of this work, the Core Ontology has been extended so that it describes the capabilities of the resources, the resource interfaces and the lifecycle information relating to the resources.

The introduction to capabilities and their modelling is first given in Chapter 4.2.1. After that, the capability model is described in Chapter 4.2.2, followed by the description of the capability associations and the capability parameters in Chapters 4.2.3 and 4.2.4, respectively. These capability associations and parameters are integral components of the capability model.

4.2.1. Introduction to capabilities

In the approach proposed here, capabilities are functionalities of resources, such as “drilling”, “milling”, “moving”, “grasping” and so on. This functionality determines the capability concept name. Capabilities have parameters, which present the technical properties and constraints of resources, such as *“speed”, “torque”, “payload”,* and so on. In other words the concept name of the capability indicates the operational functionality of the resource, whereas the capability parameters determine the range and constraints of the capability. For example, the capability with the concept name “moving” has the parameters *“velocity”* and *“acceleration”*. The capability parameters allow the determination of which resource has the capability that best fits the given product or production requirement.

Capabilities are divided into simple capabilities and combined capabilities. Combined capabilities are combinations of other capabilities, usually formed by a combination of devices, such as a robot and a gripper. A combined capability can consist of two or more simple capabilities, other combined capabilities or a collection of simple and combined capabilities. Figure 21 is a UML-diagram showing the relations between the capability-related terms.

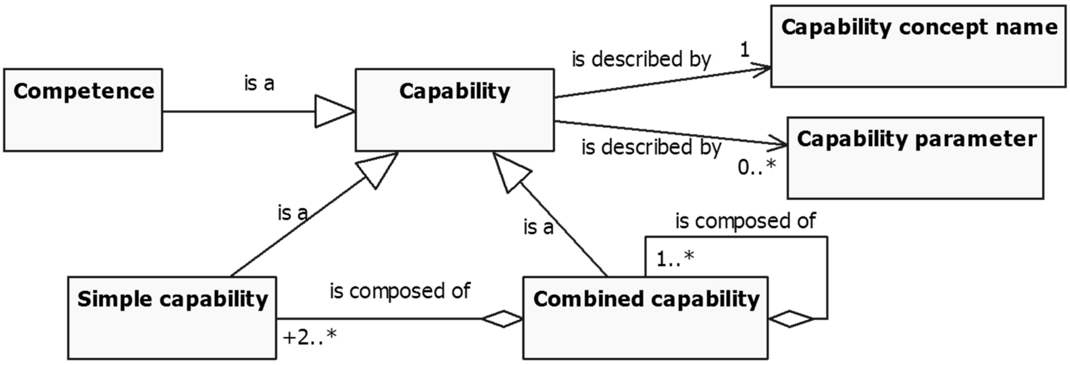


Figure 21. UML diagram of capability and related terms.

Competences are human capabilities. Because human capabilities are a vaguer concept than device capabilities they are assigned their own class. The properties of human competences cannot usually be easily measured, predicted and stated. Two persons can have the same education or training, but it doesn’t mean that they will perform the same task with the same level of performance. Experience, personality, physical shape and mental aspects affect their performance. Machines are easier to handle in this sense. In a wider context the word ‘competence’ also refers to the ability of an organization to perform processes and offer services at a certain level of performance. Even though this thesis concentrates on device capabilities, the same capability model is also intended to model human competences.

Chapter 3.4.1 discussed the ‘function-behavior-structure’ modeling framework by Gero (1990), and applied by Lohse et al. (2006a) and Kitamura et al. (2006; 2010). Function represents “what to achieve?”, whereas behavior describes “how to achieve?”. The function represents the intention of the designer, i.e. what he or she wants to achieve. The behavior represents different manufacturing methods (capabilities) that can be used to achieve the required function. For example, the required function may be “joining”, which can be achieved by several different behaviors, such as “welding”, “gluing”, “screwing”, “riveting” and so on. The structure determines the behavior of the resource. As with Kitamura et al. (2006; 2010), the approach taken in this thesis aims to describe the function and behaviour of the resources but exclude the structure level from the capability definition. The structure needs to be incorporated into the model if the model is intended to be used for programming and controlling the resources, or generating, for example, the tool paths. Therefore, given the objectives of this thesis, structure modelling is not needed.

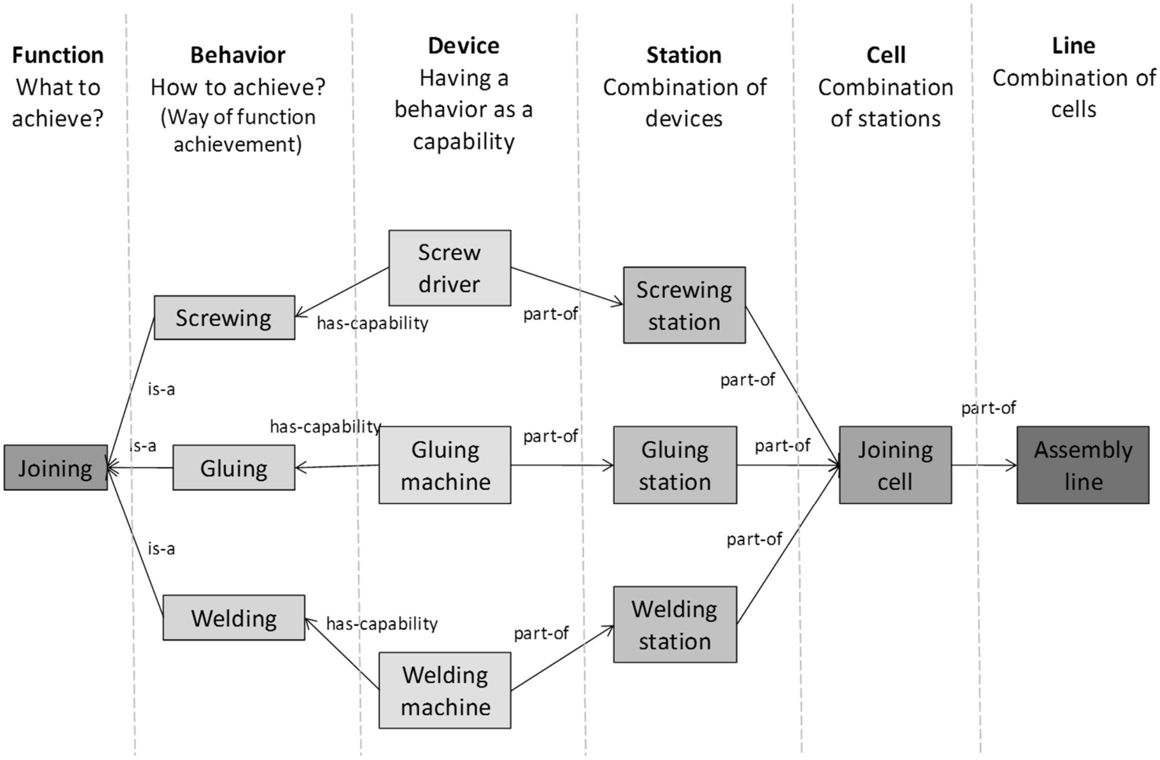


Figure 22. Relation between function, behavior and resources at different system levels.

Figure 22 shows the relation of the function-behavior framework by Kitamura et al. (2006; 2010) and the resource description approach taken in this thesis. In the proposed approach, the individual devices have capabilities, which correspond to behaviors in Kitamura's approach. The functions can be achieved by several different capabilities (behavior). The devices have capabilities (behaviors) which may be used to perform different kinds of functions. For example the "spinningTool" capability (possessed e.g. by a screw driver) can be used to perform either drilling or screwing depending on the attached tool. This means that it can be used both for the "material removing" and the "joining" functions.

It is desirable to link the functions and behaviours in order to enable the targeting of the requirements at different levels of detail. For example, the designer may want to predetermine the manufacturing method as early as the product design phase, which means that the required capability (behaviour) is fixed. On the other hand, the designer may be using the DFMA (Design for Manufacturing and Assembly) methodology, which takes the existing manufacturing and assembly technologies into account when designing the product. In this case, they may just specify that, for example, some sort of joining is required, then check the available resources capable of joining with different behaviours, and then make the detailed product design based on the available joining method. In the case of dynamic adaptation, the product can be dynamically routed based on the availability of the suitable behaviours (capabilities) which can provide the required function, leaving more freedom for resource selection. In the case of assembly processes, the required method is more often fixed by the product design than in manufacturing operations. For example, material removing of similar features can be performed by multiple different methods (capabilities).

4.2.2. Capability model

The fundamental requirement for the capability model is to be able to manage the combined capabilities of multiple co-operating resources, which can be derived from the capabilities of the individual resources involved in the combination. Therefore, the combined capabilities need to be divided into simple capabilities in a way which, firstly, supports the description of the individual resource capabilities and their emergence into combined capabilities; and secondly, supports the selection of the resources and resource combinations for a given product requirement. Basically there are two levels in the combined capability problem:

1) At the higher level, the division of the combined capabilities into simple ones and their relations need to be defined;
2) At a more detailed level, the parameters of the combined capabilities, based on the parameters of the individual capabilities, need to be reasoned out.

The proposed capability model aims for capability modularization and reusability of the capabilities among different types of resources. Generally, in modularization, the interactions between the involved components are to be minimized (Lehtonen 2007). In the case of capabilities, the division aims to minimise the dependencies between the capability parameters, so that the parameters fit naturally under one capability concept name. The capability modelling is based on the functional decomposition of upper level combined capabilities into simple capabilities and the assignation of these lower-level, simple capabilities to the devices in a modular way.

The systematic design approach of Pahl & Beitz (1996) provides a fundamental relationship between function and function decomposition. Function decomposition represents how a function is achieved through a set of sub-functions, which are finer-grained functions. (Pahl & Beitz 1996.) Similarly, the combined capabilities are achieved from finer-grained simple capabilities. Functional decomposition allows the definition of *is_part_of* relations between the simple and combined capabilities. The simple capabilities answer the questions, 'What is the purpose of each resource in the combination and what function is the resource providing for the overall system?' When multiple devices are combined, the simple capabilities form combined capabilities. The upper part of Figure 23b illustrates an example of such a division with a transportation capability; in this case with a robot unit consisting of a robot and gripper. The robot alone only has the ability to move its joints within a workspace. However, when combined with a suitable gripper, together they are able to transport pieces from one place to another.

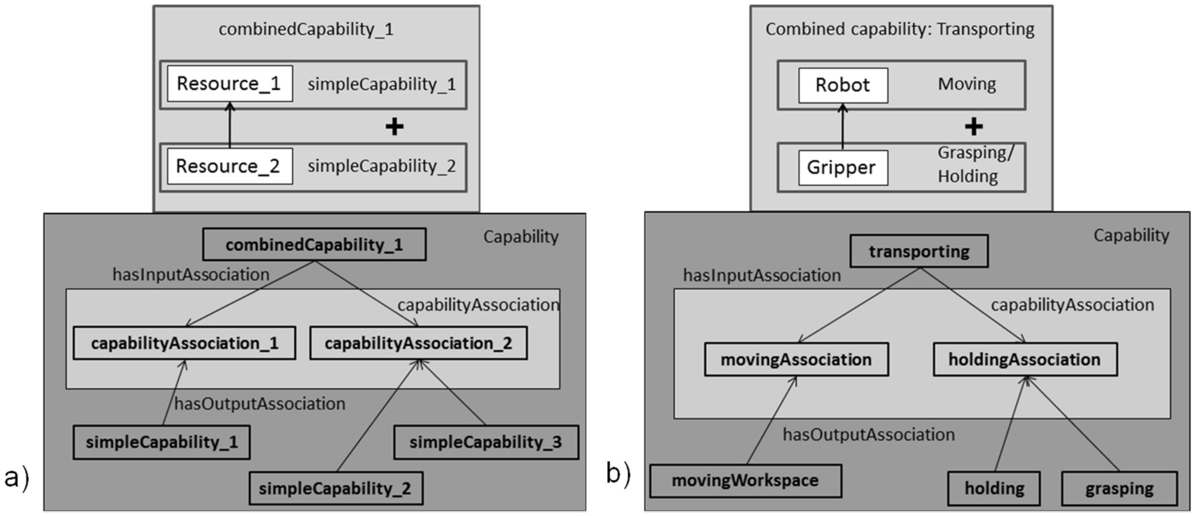


Figure 23. a) Model for combined capabilities; b) Sample of the instantiated capability model.

In the ontology, the combined capabilities are modelled using capability associations as links between the simple and combined capabilities. In the resource ontology, the devices are assigned the simple capabilities they possess. Based on the defined capability associations, the device combinations contributing to a certain combined capability can be identified and queried. Of course, the devices also need to have matching interfaces to be able to co-operate. Figure 23a represents the model for defining the combined capabilities. The same capability model can be used in different domains. Definition of the domain-specific capabilities, as well as the input and output associations for creating combined capabilities, require expert knowledge of the domain. For example, in the context of this research, production engineering knowledge is required for the definition of the capabilities used in the case studies. Figure 23b presents an example of the instantiation of the model in the production domain.

The instantiated capability model, stored in the Knowledge Base (introduced in Chapter 5.1), defines the generic capabilities, i.e. a pool of capabilities that can exist on the factory floor and can be assigned to the resources. When these generic capabilities are assigned to the resources, they become resource-specific when filled with resource-specific parameter values. The capability model can be freely extended with new capability instances when new capabilities emerge in the system, for example, when new technologies are acquired. In the next two chapters the capability associations and capability parameters will be discussed in detail.

4.2.3. Capability associations

As described in the previous chapter, and in Figure 23, the capability associations are the key to managing and presenting the combined capabilities within the ontology model. Here, the use of these capability associations will be discussed in more detail.

The principles of the capability model seen in Figure 23a are:

- Resources have simple capabilities, which provide some capability associations as their outputs.
- Combined capabilities require some capability associations as their inputs, e.g. in order for the "combinedCapability_1" to emerge, both *capabilityAssociation_1* and *capabilityAssociation_2* have to be satisfied.

- Different simple capabilities can provide the same capability association as their output, as is the case with “simpleCapability_2” and “simpleCapability_3” in Figure 23a.
- When a device or combination of devices has capabilities, which provide output for all the required capability associations (*capabilityAssociation_1* and *capabilityAssociation_2*) the combined capability (“combinedCapability_1”) emerges.
- The capability model can have multiple levels, for example the “combinedCapability_1” may provide an input association for some other combined capability. In this case the combined capability is treated as a simple capability from the upper level capability’s point of view.

An illustration of instantiating the capability model is shown in Figure 23b with the transporting capability example. In order to transport an item the system needs to be able to move within some workspace and to hold the item. Therefore the system needs both “moving” and “holding” capabilities. Holding can be implemented either by gravity (e.g. conveyor belt) or by grasping (e.g. gripper). The principles in this specific case are:

- A “transporting” capability requires *movingAssociation* and *holdingAssociation* as input.
- The *movingAssociation* means that some sort of moving capability is required.
- The *holdingAssociation* means that some sort of holding capability is needed (holding by gravity or grasping).
- A “movingWorkspace” capability provides *movingAssociation* as its output.
- “grasping” and “holding” capabilities provide *holdingAssociation* as their output.
- When a device or combination of devices has both “movingWorkspace” and “grasping” or “holding” capabilities, it has the “transporting” capability. Examples of this kind of device are either a combination of a robot and a gripper, or a conveyor alone.

Instead of building direct *is_part_of* relations between capabilities into the ontology, capability associations provide a way to present alternative capabilities which provide a common input for a certain combined capability (e.g. holding and grasping). The capability model can be extended freely upwards and downwards and new capabilities and capability associations can be added as they are learned. The capability model allows the capabilities to be modelled at different levels of abstraction. Figure 24 shows an example of “milling” and “turning” capabilities. In the case of “milling”, the “millingTool” capability is further divided into “toolHolding” and “millingCutter” capabilities, whereas in the case of “turning” the “turningTool” capability is assumed to incorporate also the “toolHolding” and “turningTool” capabilities. Another example of increasing the detail level is the finger gripper, whose capability may be either “fingerGrasping”, or the grasping can be further divided into “openFingers” and “closeFingers” capabilities. More detailed division is only needed if the capability descriptions are used for programming or real-time control of the system. Any decision about the desired level of abstraction should be made while building the related capability instances, because it will affect the selection of the suitable capability parameters for each of the capabilities involved.

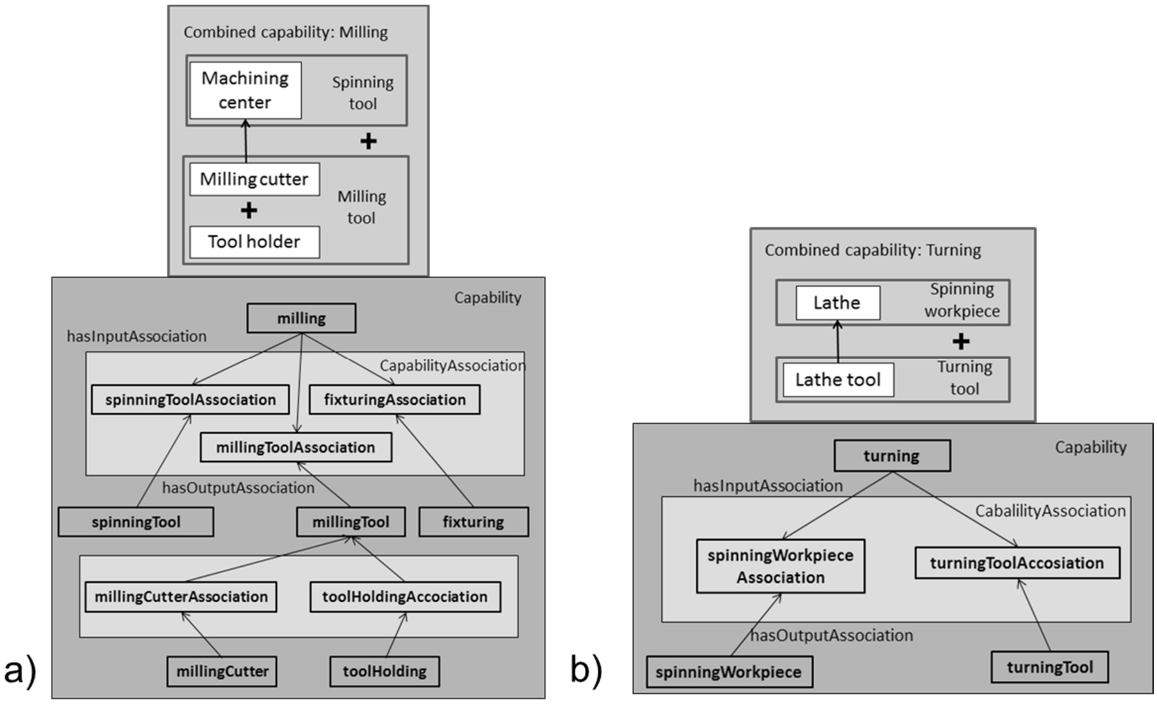


Figure 24. Modelling capabilities at different levels of abstractions, a) milling; b) turning.

Figure 25 shows more examples of the instantiated capabilities and their associations. The figure is automatically generated from the newly developed Capability Editor software tool, which will be discussed in Chapter 5.1.1. Due to the limitations of space, the figure is not complete, but extracted from a larger graph which shows all the capability instances and their associations. For example, the *movingAssociation* is an input association for many more combined capabilities than those shown in the figure. In the figure, the rectangles represent the capability associations, whereas the oval shapes represent the capabilities. The figure should be read from top to bottom. The arrows from a capability towards an association indicate the needed input associations for the combined capability, whereas the arrows from an association towards a capability indicate that the capability provides this association as its output.

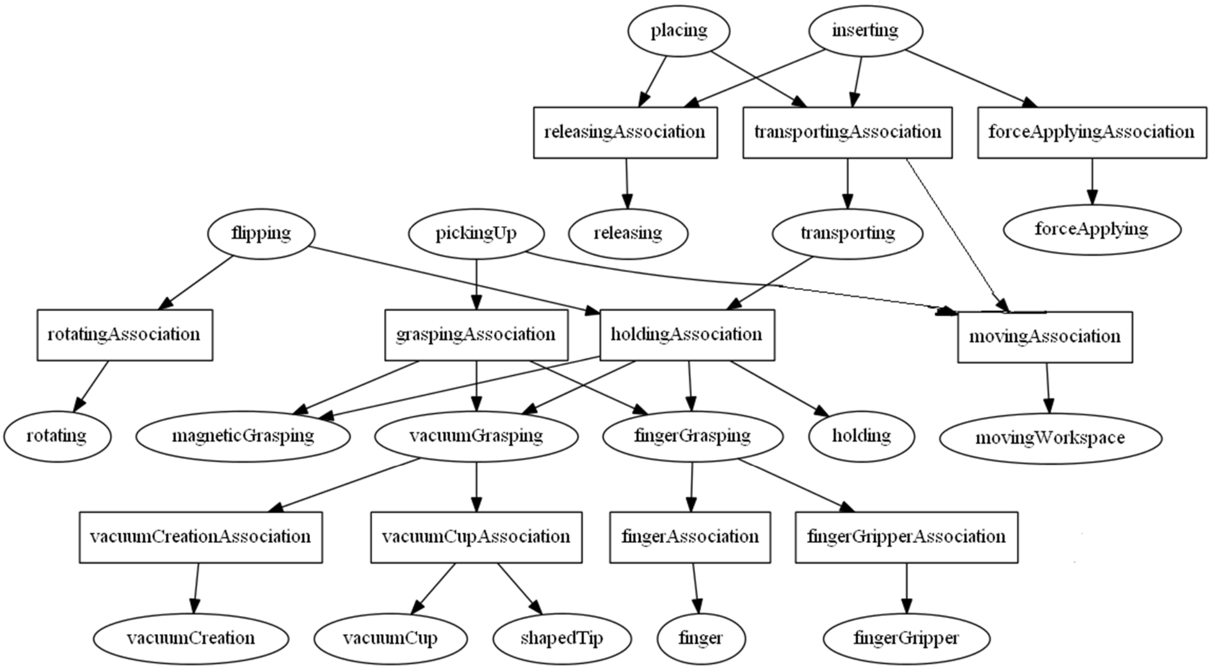


Figure 25. Small extract of the instantiated capability model.

The rest of the instantiated capabilities and their combinations through associations can be found in Appendices 1, 2 and 3. Because of the graph's size, it has been divided into smaller sections in order to make it more comprehensible. Currently, the instantiated capability model includes capabilities recognized in the use cases in the TUT-machining laboratory and the TUT-microfactory environments. These environments will be further discussed in Chapter 5. Currently, about 90 capability instances with more than 50 input and output associations are included in the capability model. This number also includes weak capabilities. If needed, there is a table in which all these capabilities are listed and explained, along with their parameters, in a material bank on the internet[1].

4.2.4. Capability parameters

Capability parameters describe the characteristics of a capability. In the capability model, the capability parameters have been defined based on the most common parameters given in the tool and machine providers' catalogs. Of course, not all the providers give the same information and not all the possible parameters found from the catalogs are included in the capability parameters. The aim was to restrict the parameters to the ones most commonly given, but to make them rich enough to allow the matching of product requirements against resource capabilities in the defined use cases. The capability model is extendable, and therefore new parameters can be added if necessary. Adding new parameters doesn't affect the structure of the capability model, but rather the rules that are used for the capability matching. If not all the parameters are available for a given device when filling in its capability information, a default value -1 can be given, to indicate that this parameter should not be taken into account in the reasoning process. In this case, the human expert needs to check the validity of the reasoning results.

The capabilities are modularized so that the redundancy and interdependencies between the parameters of the simple capabilities is minimized. This means that the parameters are assigned to

[1] http://kippcolla.serv.fi/~eeva/

the simple capabilities which they most naturally fit into. This fitting is based on production engineering domain knowledge. Also the assignment of the same parameters for multiple different capabilities is minimized. It is recognized that this "natural fit" is not so realistic in all the cases. For example, machine vision systems are so complicated that it is difficult to decide which simple capability a certain parameter belongs to. Therefore, some artificial simplifications are made during the creation of the instantiated capability model. Furthermore, for much the same reason, two logically different types of capabilities are defined:

Strong capabilities

- Strong capabilities are those resource characteristics which directly provide some kind of functionality, such as "moving", "grasping" or "releasing". Those resource properties and constraints which are naturally directly related to some simple functional capability are given as capability parameters.
- Combined capabilities are always strong capabilities. Only strong capabilities contribute to the forming of combined capabilities and therefore only they can provide the output associations or require input associations.

Weak capabilities

- Weak capabilities don't fit into the definition of capability as given in Chapter 4.2. They are those properties and constraints of resources that do not naturally directly relate to any simple functional capability, but which provide important input information when selecting resources for a specific application. In practice, they are additional parameters which supplement the strong capabilities to aid in the decision making, such as "basicDeviceInfo" (containing parameters for describing the device dimensions and weight). Even though they don't fit into the logical definition of capability, weak capabilities are treated in the ontology model in the same way as strong ones. This reflects their importance in resource selection.
- Some parameters are common to multiple different capabilities and resources. For example, there may be a restriction on the minimum or maximum size of an item that can be processed with different resources. Instead of assigning these parameters separately to each capability they may restrict, it is easier to have one weak capability (in this case "minItemSize" or "maxItemSize") that can be assigned to all the relevant resources.
- Sometimes, weak capabilities are needed to make a detailed match between product requirement and resource capabilities. For example, the strong capability "movingWorkspace" (indicating the resource has a capability to move within a workspace) can be supplemented with the "degreesOfFreedom" and "workspaceType" weak capabilities.
- As weak capabilities don't provide any functionality, they don't contribute to forming any combined capabilities. Therefore, they don't provide any output associations or require any input associations. Hence, they don't appear in the graphs showing the associations between the capabilities found in Appendices 1, 2 and 3.

Table 6 and Table 7 show examples of typical resource characteristics given in the resource provider catalogs and their allocation to the simple capabilities in our capability model. Two separate use cases are considered: robot and gripper, and milling machine with related tools. The shaded columns show the typical parameters of the devices, while the white columns indicate which simple capability this parameter is attached to.

Table 6. Typical resource parameters and their link to the simple capabilities – Case robot and gripper.

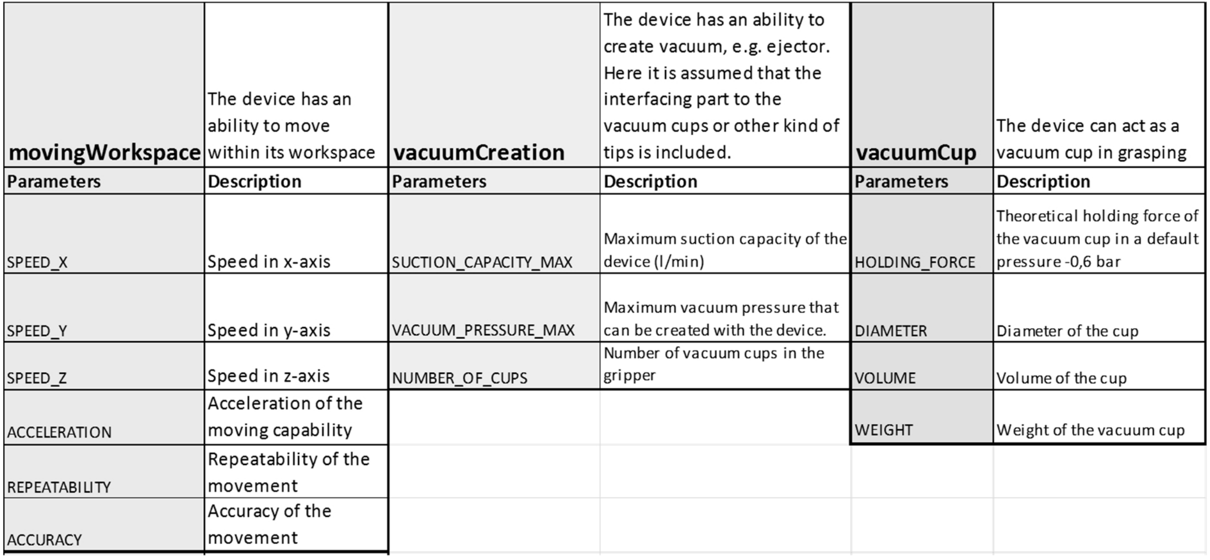
Robot params	**Attached to capability**	**Vacuum gripper params**	**Attached to capability**	**Finger gripper params**	**Attached to capability**
dimensions	*basicDeviceInfo*	**Gripper**		**Gripper**	
weight	*basicDeviceInfo*	dimensions	*basicDeviceInfo*	dimensions	*basicDeviceInfo*
workspace (type)	*workspaceType*	weight	*basicDeviceInfo*	weight	*basicDeviceInfo*
DOF	*degreesOfFreedom*	Suction capacity	*vacuumCreation*	grasping force	*fingerGripper*
workspace (volume)	*workspaceType*	Vacuum pressure	*vacuumCreation*	stroke	*fingerGripper*
payload	*payload*	Number of cups	*vacuumCreation*	payload	*fingerGripper*
velocity (in direction)	*movingWorkspace*	**Vacuum cups**		opening/closing time	*fingerGripper*
acceleration	*movingWorkspace*	dimensions	*basicDeviceInfo*	accuracy	*fingerGripper*
accuracy	*movingWorkspace*	weight	*basicDeviceInfo*	repeatability	*fingerGripper*
repeatability	*movingWorkspace*	vacuum cup diameter	*vacuumCup*	number of fingers	*fingerGripper*
force (in direction)	*forceApplying*	vacuum cup volume	*vacuumCup*	type	*fingerGripper*
working conditions (max/min temperature,	*environmentProperties*	holding force	*vacuumCup*	**Fingers**	
				Tolerance	*finger*
				Type	*finger*

Table 7. Typical resource parameters and their relations to the capabilities – Case milling machine and tools.

Milling machine params	**Attached to capability**	**Milling tool params**	**Attached to capability**	**Fixture params**	**Attached to capability**
Drive power	*spinningTool*	**Milling cutter**		Dimensions	*basicDeviceInfo*
Spindle speed range	*spinningTool*	Dimensions	*basicDeviceInfo*	Weight	*basicDeviceInfo*
Rapid traverse speed	*movingWorkspace*	Weight	*basicDeviceInfo*	Max spread	*fixturing*
Max spindle torque	*spinningTool*	Tool type	*millingCutter*	Min spread	*fixturing*
Cross travel X	*workspaceBox*	Insert shape	*millingCutter*	Max item weight	*fixturing*
Vertical travel Y	*workspaceBox*	Cutting edge angle	*millingCutter*	Position of workpiece	*fixturing*
Longitudinal travel Z	*workspaceBox*	Cutting edge length	*millingCutter*	Number of workpieces	*fixturing*
Degrees of freedom	*DOF*	Diameter of tool	*millingCutter*	Max/min workpiece size	*itemMaxSize/item MinSize*
Accuracy	*movingWorkspace*	Version	*millingCutter*		
Repeatability	*movingWorkspace*	Nose radius	*millingCutter*		
Distance between spindle center and table	*millingProperties*	Number of teeth	*millingCutter*		
Distance between spindle nose and pallet	*millingProperties*	Suitable materials	*millingCutter*		
Spindle diameter	*millingProperties*	Tool entry (min/max)	*millingCutter*		
Spindle length	*millingProperties*	Holder interface diameter	*millingCutter*		
Payload (product +	*payload*	**Tool holder**			
Max workpiece size	*itemMaxSize*	Dimensions	*basicDeviceInfo*		
Size	*basicDeviceInfo*	Weight	*basicDeviceInfo*		
Weight	*basicDeviceInfo*	Tool diameter	*toolHolding*		
Workpiece positioning accuracy	*indexing*	Tool entry (min/max)	*toolHolding*		
Number of tools	*toolChanging*	Effective length	*toolHolding*		
Tool changing time	*toolChanging*				

While Tables 6 and 7 present the parameters from the device point of view, Table 8 shows a few examples of the capabilities and their parameters from the capability model's point of view. The shaded columns show the name of the capability concept and its related parameters. The white columns give their definitions for them. The rest of the instantiated capabilities (currently around 90 capability instances) in the capability model and their parameters can be found from the internet page[2] .

Table 8. Capabilities and their parameters – Examples: "movingWorkspace", "vacuumCreation" and "vacuumCup".



movingWorkspace	The device has an ability to move within its workspace	**vacuumCreation**	The device has an ability to create vacuum, e.g. ejector. Here it is assumed that the interfacing part to the vacuum cups or other kind of tips is included.	**vacuumCup**	The device can act as a vacuum cup in grasping
Parameters	**Description**	**Parameters**	**Description**	**Parameters**	**Description**
SPEED_X	Speed in x-axis	SUCTION_CAPACITY_MAX	Maximum suction capacity of the device (l/min)	HOLDING_FORCE	Theoretical holding force of the vacuum cup in a default pressure -0,6 bar
SPEED_Y	Speed in y-axis	VACUUM_PRESSURE_MAX	Maximum vacuum pressure that can be created with the device.	DIAMETER	Diameter of the cup
SPEED_Z	Speed in z-axis	NUMBER_OF_CUPS	Number of vacuum cups in the gripper	VOLUME	Volume of the cup
ACCELERATION	Acceleration of the moving capability			WEIGHT	Weight of the vacuum cup
REPEATABILITY	Repeatability of the movement				
ACCURACY	Accuracy of the movement				

Table 8 presents the simple capabilities which lead to the emergence of the combined capability "transporting". A robot typically has the "movingWorkspace" capability, whereas a gripper's "vacuumGrasping" capability can be enabled by the "vacuumCreation" and "vacuumCup" capabilities. In addition to those capabilities, other capabilities may also be assigned to the resources. For example, in the case of a robot and gripper, they both have at least the "basicDeviceInfo", and the robot has "payload", "degreesOfFreedom" and "workspaceType" capabilities. Based on the parameters of the simple capabilities assigned to the resources, it is possible to make a match between the product requirements and the resource capabilities. Often, this matchmaking requires some adjustment of the parameters, which is why the combined capability rules for combining capability parameters have been defined. For example, the maximum weight of the product that can be transported by the robot and the vacuum gripper is estimated as follows:

> *Robot payload minus gripper weight (gripper includes vacuum creator and vacuum cups) OR Gripper payload calculated based on the holding force and amount of vacuum cups in the gripper. The lower value has priority.*

The parameters for the combined capabilities are reasoned according to the properties of the devices involved in the combination. In other words, they are based on the parameters of their simple capabilities. In addition to those simple capabilities that enable the emergence of combined capabilities at the capability concept name level, there are other simple capabilities assigned to the device which may be taken into account in the detailed matchmaking. For instance, in the above example the weak capability "basicDeviceInfo" is needed in order to know the weight of the gripper and to be able to subtract it from the robot payload. With the capability model alone, it is only possible to merge the capability concept names. Detailed-level reasoning with capability parameters and capability matching is enabled by the rule-base discussed in Chapter 4.4.4.

[2] http://kippcolla.serv.fi/~eeva/

4.3. Components of the overall resource description

The overall resource description utilizes the capability model discussed in the previous chapter for describing the resources and their functionality. In addition to the capability information containing the capability concept name and the capability parameters, the collected lifecycle information, business information and interface description are also part of the overall resource description, as shown in Figure 26. The capabilities have a reference to the capability taxonomy described in Chapter 4.4.1. The capability taxonomy includes only functional, i.e. strong, capabilities, and therefore only the strong capabilities refer to the taxonomy.

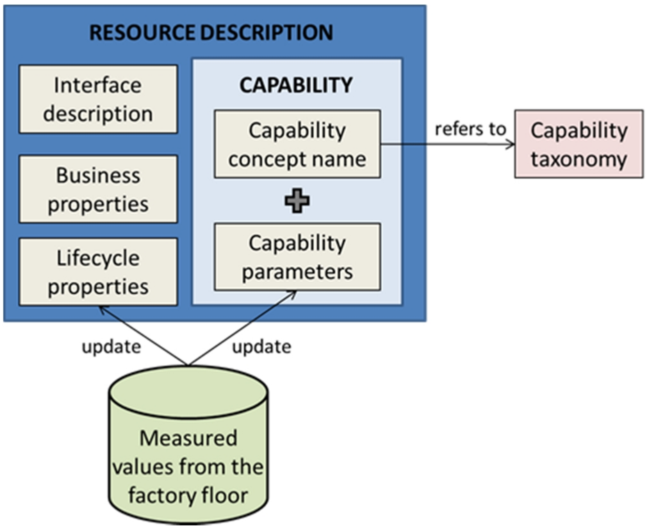


Figure 26. Components of the resource description.

As has been said, a production environment is constantly changing, and the condition and capabilities of the resources evolve during their individual lifecycles and usages. Therefore, the description of the resource has to be updated over time. For this reason, devices have two separate, but linked representations within the ontology: device blue prints and individual devices. The device blue print describes the capabilities and properties of one type of device, as given in the suppliers' catalogs. This is the nominal capability of the device. The individual devices are presented in a separate class which refers to the blue print device, yet presents the actual capabilities of the particular, individual resource which exists on the factory floor. The individual devices have actual capabilities, which are affected by the lifecycle of each individual device and updated according to measured values from the factory floor. For example, if the measured accuracy of the machine differs from the value defined in the nominal capability, this updated value can be given in the actual capability definition. Maintenance and service operations or adaptations done to the resource can also change its capability. Figure 27 shows the relations between the device blue prints and the individual devices, and their associated information elements.

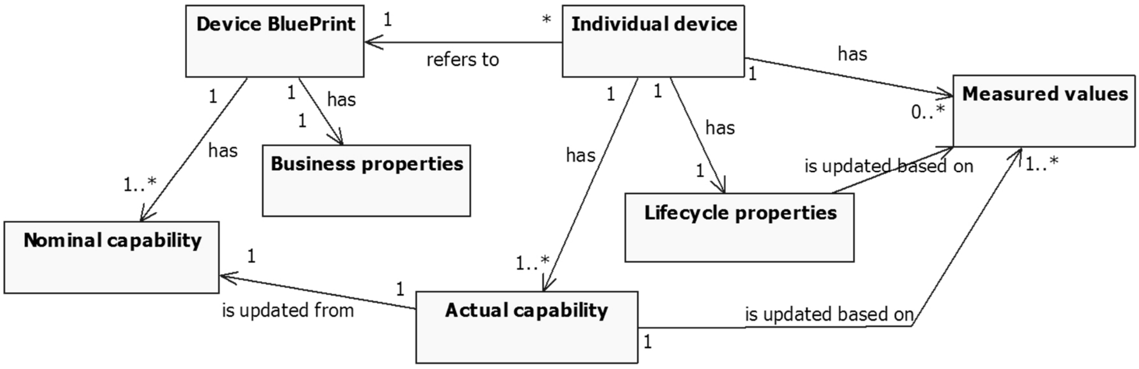


Figure 27. Device blue prints and individual devices.

If there are a number of similar machines on the factory floor, these individual devices will all have a reference to the same device blue print. This device blue print defines the nominal capability of these devices. If there are no measured values or other evidence of the actual capability of an individual device available, the capability of the individual device is assumed to be the nominal capability of the referenced device blue print. This is the case, for example, when a brand new device is taken into use. This approach means the capability definitions can be reused, and allows for the accuracy of the definition to be enhanced as more data becomes available.

4.3.1. Business and lifecycle properties

Business properties is included as part of the overall resource description in order to provide more information to support resource selection. This static business information is given as a set of parameters for the device blue prints. These parameters relate mainly to the costs of acquiring and using the devices. Table 9 explains the parameters included in the business properties. The business properties, as well as the lifecycle properties, are put into the ontology in the same way as the capabilities. Logically, they are very similar to the weak capabilities.

Table 9. Business properties – Description of the parameters.

Parameters	Description
RENTAL_COST	Cost of renting the device
PURCHASE_COST	Cost of purchasing the device
SUPERVISING_OPERATORS	Number of operators needed to supervise or operate the device
AVERAGE_LIFETIME_IN_TIME	Estimated average lifetime of the device in time
AVERAGE_LIFETIME_IN_CYCLES	Estimated average lifetime of the device in operation cycles
SETUP_TIME	Average time needed for setting up the device for a new process
INSTALLATION_TIME	Average time needed to install the device in a new configuration
CONSUMABLE_COST	Estimated cost of consumables (e.g. oil) needed to run the device
ENERGY_CONSUMPTION	Estimated energy consumption of the device in operation.
PRODUCTION_CAPACITY	Estimated production capacity of the device

As shown in Figure 27, the representation of the individual devices includes the collected and measured lifecycle information of the device, which can be used in the planning process for re-use and adaptation. In practice, the raw lifecycle and operational data is collected on a separate resource database and, at certain intervals, the filtered and relevant key figures, such as Mean Time Between Failure (MTBF), Mean Time to Repair (MTTR), maintenance costs and operation time, can be saved as part of the overall resource description. The lifecycle properties contain most of the parameters from the business properties and other relevant parameters, and these are updated during the individual device's lifecycle based on the values obtained from the factory floor. These dynamic lifecycle properties are assigned to the individual devices and are therefore resource-specific. Table 10 explains the parameters of the lifecycle properties.

Table 10. Lifecycle properties – Description of the parameters.

Parameters	Description
PURCHASE_COST	Cost of purchasing the individual device
RENTAL_COST	Cost of renting the individual device
INSTALLATION_TIME	Average time needed to install the individual device into a new configuration
SETUP_TIME	Average set-up time of the individual device
MTBF	Mean time between failures of the individual device
MTTR	Mean time to repair the individual device
MAINTENANCE_COST	Average maintenance cost of the individual device
MAINTENANCE_TIME	Average time needed for maintenance of the individual device
SERVICE_INTERVAL	Average interval for service operations for the individual device
SERVICE_TIME	Average time needed for service operations
PROCESS_SCRAP	Average scrap produced by the individual device
CONSUMABLE_COST	Cost of the consumables (e.g. oil) needed to run the individual device
ENERGY_CONSUMPTION	Average energy consumption of the individual device in operation.
PRODUCTION_CAPACITY	Average production capacity of the individual device
OPERATING_TIME	Time the individual device has been in operation
OPERATING_CYCLES	Amount of operation cycles the individual device has been operating

Figure 28 shows the relevant lifecycle aspects which affect the capability of the resource and the selection of a suitable resource for a certain application. The resource database is constantly recording the runtime data of the resources on the factory floor, such as the operational values. The information on how a specific resource or system performed in a specific process, while processing a specific product, can later on be used for the resource selection for similar products. The resource's condition, its remaining lifetime and consequently its capability will vary depending on the applications it has been used for, the environments it has been used in, the processes it has performed and the process parameters which have been used. Different statistical methods from the area of maintenance research could be used to estimate these parameters. It is planned to handle this context-specific historical information with a role engine, whose development is beyond the scope of this study. However, the role engine will be briefly discussed in Chapter 6.3, under Future Work.

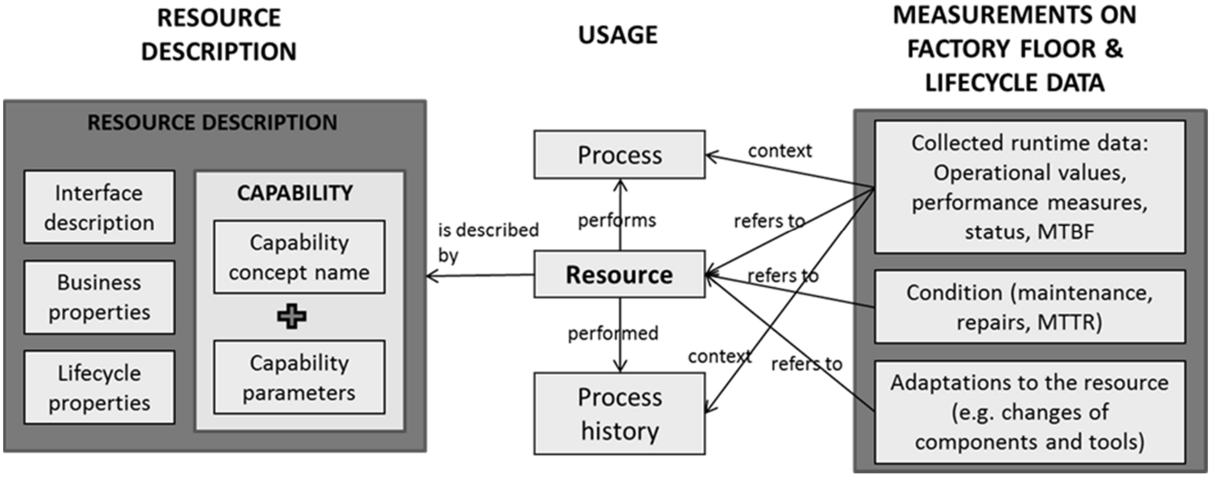


Figure 28. Relevant lifecycle aspects affecting the resource capability and the selection of resources for a specific application.

4.3.2. Interface description

The interface description is seen as a crucial part of the overall resource description. It is needed to enable the automatic creation of resource combinations when searching for resources having suitable capabilities for the given product order. Standardized interface descriptions have been widely studied in, for example, the EUPASS project (Siltala et al. 2008). Therefore, it is unnecessary to develop a detailed interface description for this thesis. The interface description, which is adapted from the EUPASS interface definitions and integrated into the Core Ontology, can be seen in *Figure 29*. It is simplified, but provides enough expressiveness for the resource compatibility matching process.

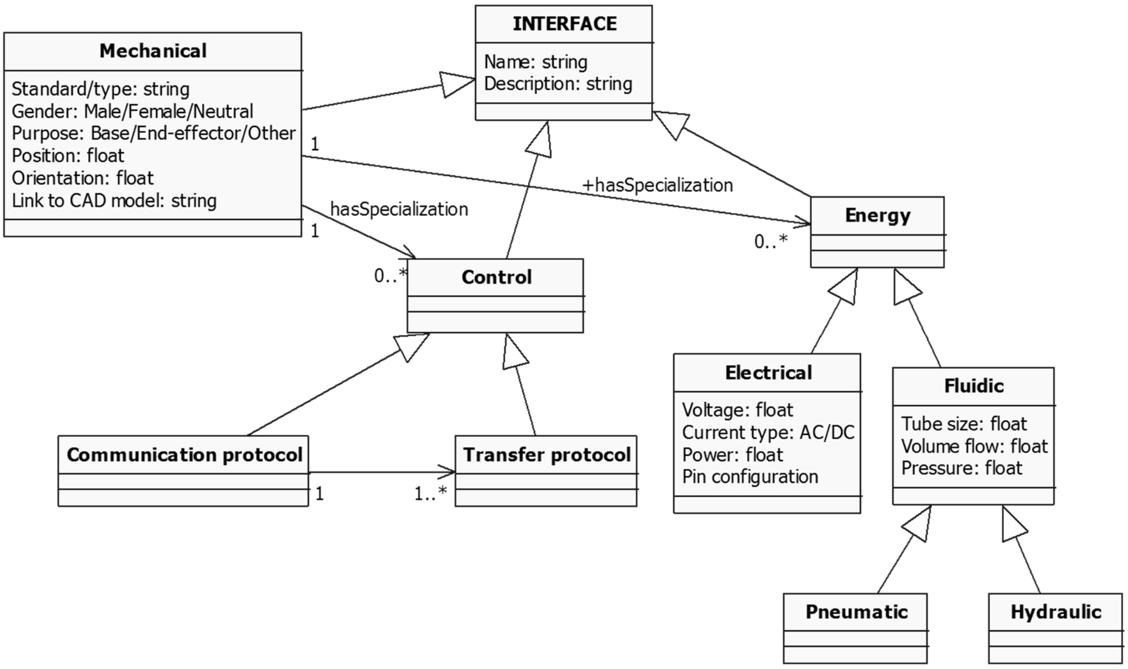


Figure 29. Resource interface description.

Some explanations of the interface description:

- The purpose of the mechanical interface always has to be defined. It can be either: “Mounting to the base”, “End-effector mounting” or “Attaching to other devices”.
- Interfaces nearly always have some sort of physical part, and therefore all the interfaces are defined through mechanical interfaces. By using the “hasSpecialization” property, the mechanical interface can refer to control or energy interfaces. The connector of the interface is specified by the mechanical interface standard definition. This definition of the mechanical interface supports the position and orientation definition for the energy and control interfaces.
- The communication protocol defines which communication protocol is used in the control interface. The communication protocols describe the set of rules to be used in communication exchange. They have their own syntax, semantics and synchronization rules. Common protocols are, for example, Modbus, Profibus and CIP (Common Industrial Protocol). The communication protocol operates over one or more physical layers, which are specified by the Transfer protocol. Typical examples of the transfer protocols are RS485, USB, Ethernet, fiber, WLAN.

For the devices, several interfaces can be defined according to their purpose. The interfaces are given either as input or output interfaces in order to facilitate the device compatibility checking. Input interfaces are those which need to be attached in order for the device to become functional. For example, a tool has an input interface for a tool holder. A tool holder, on the other hand, has an output interface for a tool and an input interface for the machine (e.g. milling or turning machine).

4.4. Framework for capability-based matching of product requirements and system capabilities

The core of this adaptation methodology lies in the capability-based matching of product requirements and system capabilities. Figure 30 presents the framework for this capability-based matching. As defined in Chapter 4.1, the required processes should be expressed in such a way that they are closely related to the formalism that describes the available resource capabilities, and vice versa. The capability taxonomy has been developed for this purpose, as shown in Figure 30. The matching is performed according to the capability taxonomy and capability matching rules which connect the product and resource domains to each other. The taxonomy, included in the Core Ontology, is used to make a crude search that maps the resources with the required capabilities at a high level. The detailed reasoning with the capability parameters and combined capabilities is based on rule-based reasoning, as will be explained in the following chapters.

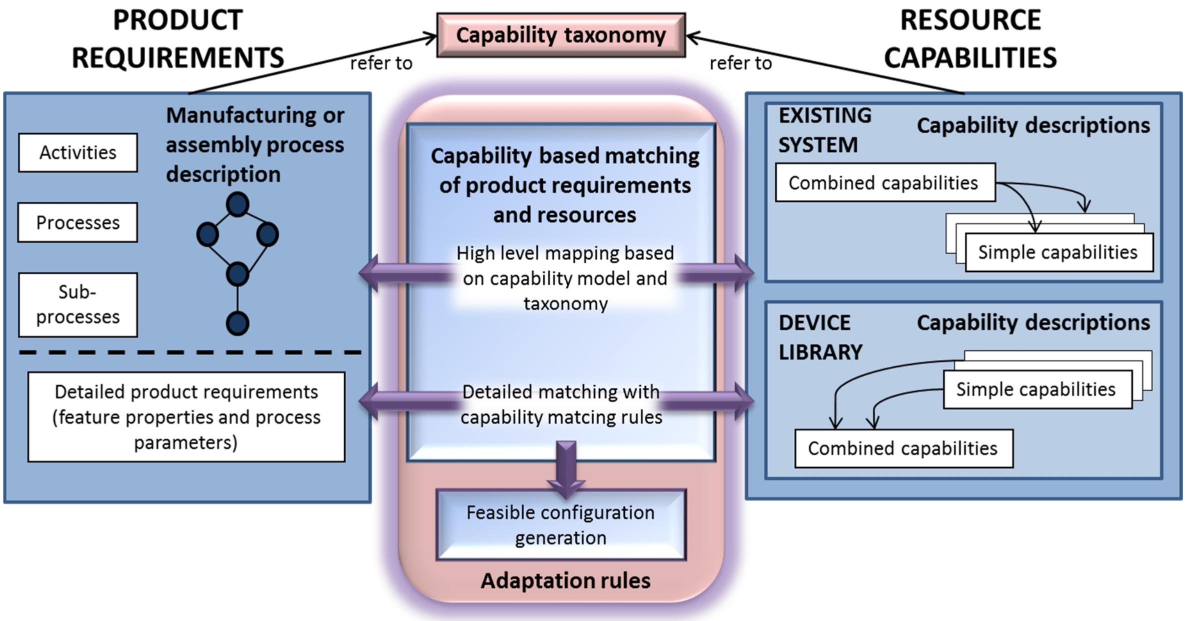

PRODUCT REQUIREMENTS
Capability taxonomy
RESOURCE CAPABILITIES
refer to
refer to
Manufacturing or assembly process description
Activities
Processes
Sub-processes
Detailed product requirements (feature properties and process parameters)
Capability based matching of product requirements and resources
High level mapping based on capability model and taxonomy
Detailed matching with capability matcing rules
Feasible configuration generation
Adaptation rules
EXISTING SYSTEM
Capability descriptions
Combined capabilities
Simple capabilities
DEVICE LIBRARY
Capability descriptions
Simple capabilities
Combined capabilities

Figure 30. Simple framework for matching the product requirements with the resource capabilities and forming a feasible system configuration under the adaptation rules, modified from (Järvenpää et al. 2011a).

The capability-based matching follows the principles of Service Oriented Architecture (SOA). In SOA, the services need to be described to the other parties in the system in a way which allows the matching of the provider's description of its offerings and the requestor's description of its needs. This entails the services being described through their functional (what can they do?), behavioral (how is the functionality achieved?) and non-functional (constraints of the previous two) aspects. The resource description and capability model facilitate the description of the behavior, whereas the capability taxonomy allows the capabilities to be linked to the related function. The non-functional aspects are handled through the capability parameters and, in some cases, the weak capabilities.

Figure 31 shows the concept of the SOA-based matching of the products and resources. The product can be seen as a service requestor, whereas the order (including the product requirements) is seen as a service request. Resources are service providers, which provide the services through their capabilities. The role of the service broker is taken by the ontology, which describes the product requirements and resource capabilities, the capability model, and the rule-base, thus facilitating the matching between the order and the offerings.

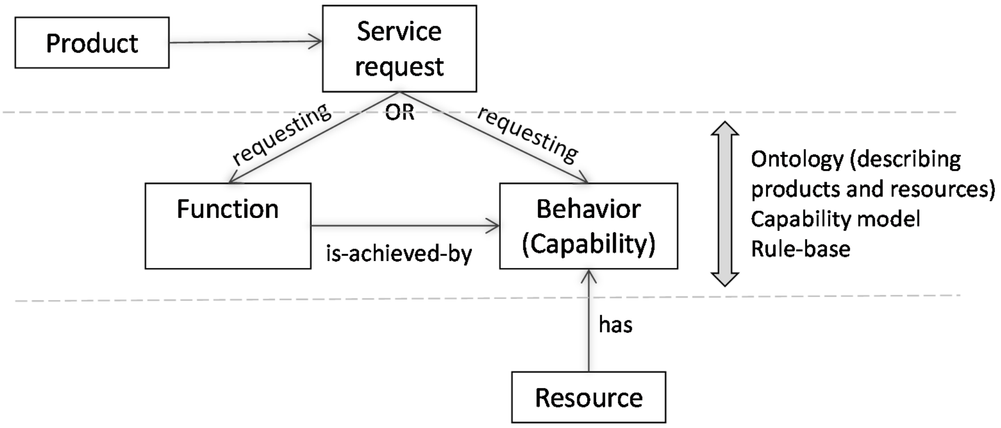


Figure 31. SOA-based architecture for product-resource matching.

Depending on the order, if the manufacturing/assembly plan is predefined, certain behaviors (capabilities) may be directly requested. If these are not detailed, higher level functions may be requested. This means that the request can be targeted at different levels in the capability taxonomy, as discussed in the next chapter. If the request is targeted at functions, then all the devices having the capabilities which can achieve that function will be offered. Functionality is the most essential aspect of the service, whereas the non-functional properties merely constrain the selection of the service, given that the functionally suitable candidates have been identified.

Chapter 4.4.1 will first introduce the formed capability taxonomy, followed by a discussion about how the product requirements are defined in Chapter 4.4.2. The high level capability mapping is then discussed in Chapter 4.4.3. The rule-base and the detailed capability matching carried out based on the rules are discussed in Chapters 4.4.4 and 4.4.5 respectively.

4.4.1. Capability taxonomy

The capability model itself neither separates nor links the behavior and function discussed in Chapter 4.2.1. It is the capability taxonomy which makes the link between these two. As stated by Kitamura et al. (2006), a general function decomposition tree includes possible alternative ways of function achievement in an 'OR' relationship for a specific goal function. The generic ways of achieving a function are, in turn, organized in '*is-a'* relations, according to their principles. (Kitamura et al. 2006.) The capability taxonomy defines these alternative ways of achieving a function in a hierarchical tree defining the general – specific (*is_a)* relations between the capabilities at different levels of detail, i.e. functions and behaviors. This hierarchy supports flexible, subsumption-based mapping of the required functions and provided capabilities.

The taxonomy defines the possible functional capabilities that can be requested by the products. The main purpose of the taxonomy is to provide the link between the processes and the capabilities, and thus to map the product requirements and resource capabilities at the concept name level. Classification of the capabilities into the form of a taxonomy enables the fast discovery of candidate capabilities. However, the class of a capability gives only a rough picture of what it actually can do. To really know what the capability means, it needs to be extended with the parameter information, as was done in the capability model.

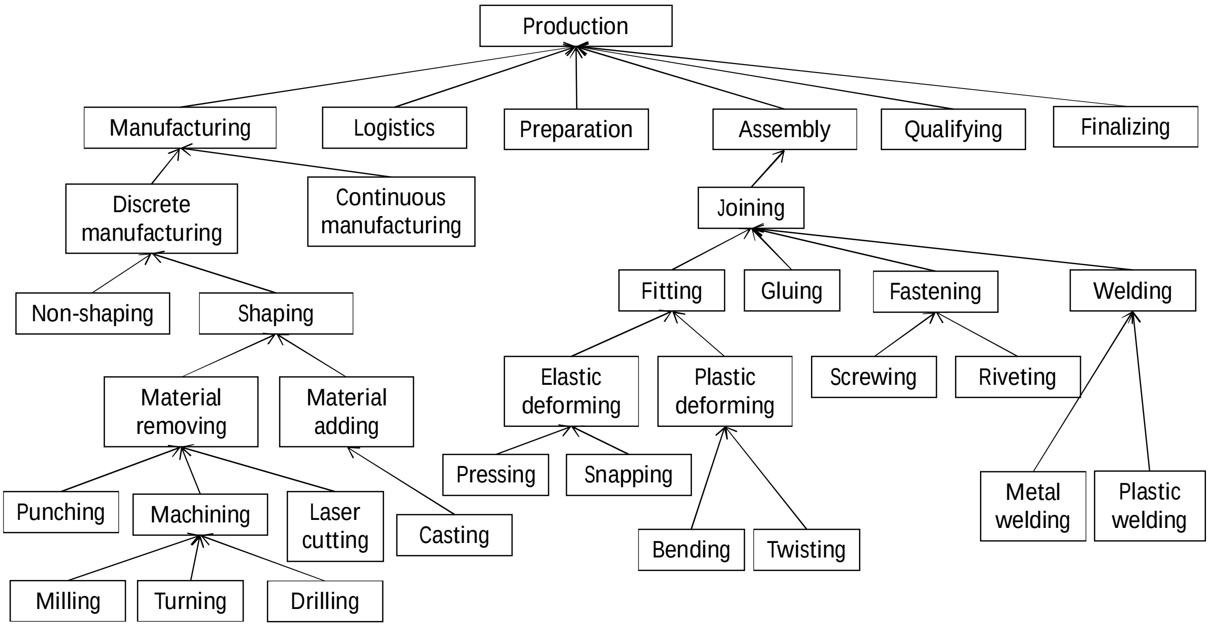


Figure 32. A small part of the production capability taxonomy included in the ontology.

Figure 32 presents a small part of the implemented capability taxonomy as an example. The full taxonomy can be found in Appendix 4. It is adapted from multiple existing taxonomies and process categories, which include the EUPASS processes (Lohse et al. 2008), the production taxonomy used in the CO2PE! -initiative (CO2PE! 2010), the Wikipedia manufacturing taxonomy (Wikipedia 2011), and the German standard DIN 8580, which provides a classification of manufacturing methods. The EUPASS process ontology (Lohse et al. 2008) has been utilized in particular for the assembly, logistic, preparation and finalization parts of the taxonomy, while the others have influenced mainly the manufacturing side.

The taxonomy is implemented with the possible case applications in mind and adapted to include everything that is needed in the case studies. It doesn't aim to cover the full production area or claim to be complete. The definition of a more complete taxonomy would require the wide participation of the production community, and would still not satisfy every participant's requirements. The taxonomy built for this thesis is defined to be expressive enough for real system design tasks and for demonstrating the proposed adaptation and capability-matching methodology. The taxonomy is static while it is being used. However, it can be easily extended based on emerging needs, for example, when applying the methodology to different domains or when new capabilities are needed.

The taxonomy is built into the Core Ontology, in order to allow direct relations between the process descriptions and the taxonomy, as well as between the resource capabilities and the taxonomy. The hierarchy between the taxonomy instances is created using a *hasParentNodeInTaxonomy* relation, by which the lower level capability categories are subsumed into the upper level categories.

4.4.2. Definition of the product requirements

Product requirements are those product characteristics or features which require a set of processes to be performed in order for the product to be transformed towards the finished product. These processes are executed by the devices and combinations of devices possessing adequate functional capabilities. The product requirements can be expressed in terms of the required capabilities and their temporal and logical order. In the proposed approach, the product requirements are expressed in the form of a pre-process plan, generated by a tool called Pro-FMA Extended (Garcia et al., 2011), which will be introduced in more detail in Chapter 5.1.2. The pre-process plan describes the required capabilities at a generic level, such as “material removing”. Each of the process steps in the pre-process plan has a reference to a certain level in the capability taxonomy. The pre-process plan is linked to the product information and the activities in the pre-process plan are linked to a specific feature, or part information, in the ontology. This way, the specific characteristics of a product or part, such as size, weight, geometry and so on, can be used as an input for the reasoning when searching for suitable resources. They have a significant effect on, for example, gripper selection and feeding methods.

4.4.3. High-level capability mapping

Together with the developed capability model, the capability taxonomy integrated into the Core Ontology allows mapping between the product requirements and the resource capabilities at the capability concept name level. As discussed in the previous chapter, the activities in the pre-process plan for the product refer to specific levels in the capability taxonomy. Naturally, the capability instances also refer to a certain level in the capability taxonomy, thus enabling the link between products requesting the capabilities and the resources providing the capabilities.

The taxonomy level to which the pre-process plan refers depends on how detailed the available information is regarding the required or desired processing methods. For example, the product designer may have defined that a specific joining method, such as riveting, should be used to join two parts together. Alternatively, they may have only defined that some sort of joining method is required, leaving the possibility for the joining method to be determined based on the capabilities available on the factory floor. In the former case, the product requirement is targeted at the particular method in the capability taxonomy, whereas in the latter case the requirement is targeted at the joining level in the taxonomy. The pre-process plan defined by the Pro-FMA Extended software always expresses the activities at a very generic level.

The capability taxonomy enables the capabilities of the resources to be connected with the capabilities required by the product at different levels of detail. It allows different devices that are able to perform the same function (e.g. “material removing”) using different behaviors (e.g. “milling”, “turning”, “drilling”, etc.) to be searched for. The parameters of the capabilities will then determine if the suggested device is able to fulfil the given requirements. For example, if the requirement is [material removing, hole of diameter 20mm and depth 50mm, aluminium], the parameters of the capabilities which are a subset of the material removing capability (e.g. “milling”, “turning” and “drilling”) will then express which device combination is able to provide the required removal of material within the required parameters. Capability matching rules, discussed in the next section, will be used to match the parameters.

As the devices are assigned the simple capabilities they possess, based on the defined capability associations in the capability model, the device combinations which contribute to a certain combined capability can be identified and queried with SPARQL RDF query language. In the same way, it is possible to reason out the capabilities that the resource combinations have. Figure 33 illustrates how the capability associations (inputs and outputs) are used to make a match between the capabilities existing with the current resources and the required capabilities at the concept name level. By using the associations, it is possible to answer, for instance, the following questions: “Which devices need to be combined in order to achieve a certain combined capability? “What combined capabilities can a certain combination of devices have?”

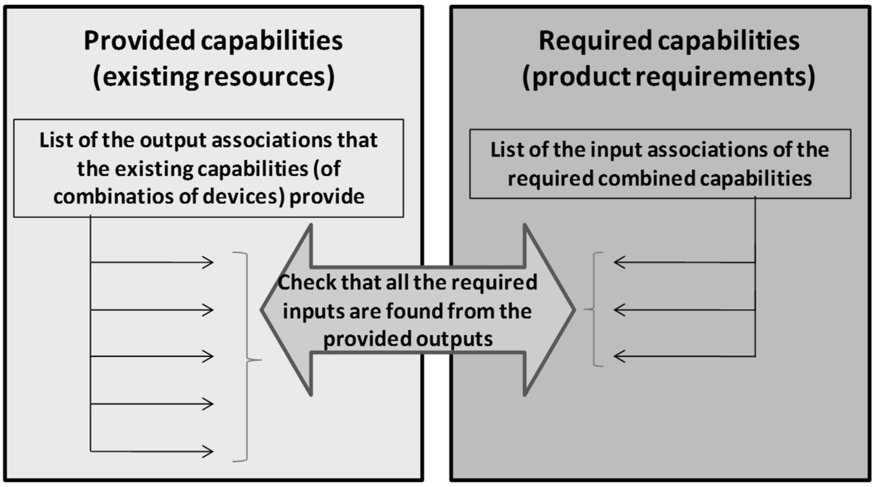

Provided capabilities (existing resources)
List of the output associations that the existing capabilities (of combinatios of devices) provide
Required capabilities (product requirements)
List of the input associations of the required combined capabilities
Check that all the required inputs are found from the provided outputs

Figure 33. Matching of capability output and input associations (Järvenpää et al. 2012a).

Figure 34 gives a practical example of the input and output association matching. The input and output associations are written in italics (e.g. *spinningTool*). The white rectangles represent the resources which possess those capabilities that provide the shown associations. As seen in this figure, the combined capabilities are formed by climbing hierarchically from lower level combinations to upper ones, e.g. the tool holder and threading cutter combination is viewed as a threading tool on the next level.

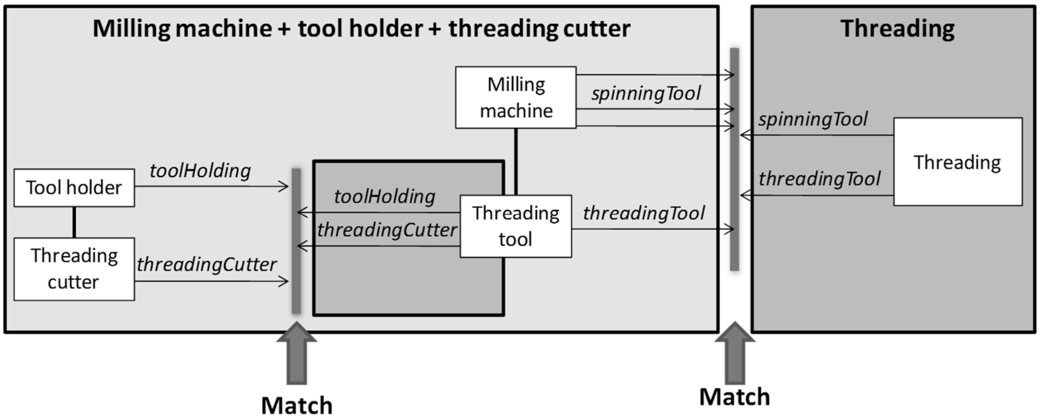


Figure 34. Example for matching the capability input and output associations (Järvenpää et al. 2012a).

The example shows that, based on the capability taxonomy and the capability model, it is possible to map the product requirements with the resource capabilities at the concept name level. This is called high-level capability mapping. The capability associations only provide a means to manage the combined capabilities at the capability concept name level, not at the parameter level. Therefore, in order to make a detailed match between the required and provided capabilities, more intelligent reasoning is required. For this purpose, rule-based reasoning is applied, as discussed in the following two chapters.

4.4.4. The rule-base for capability matching

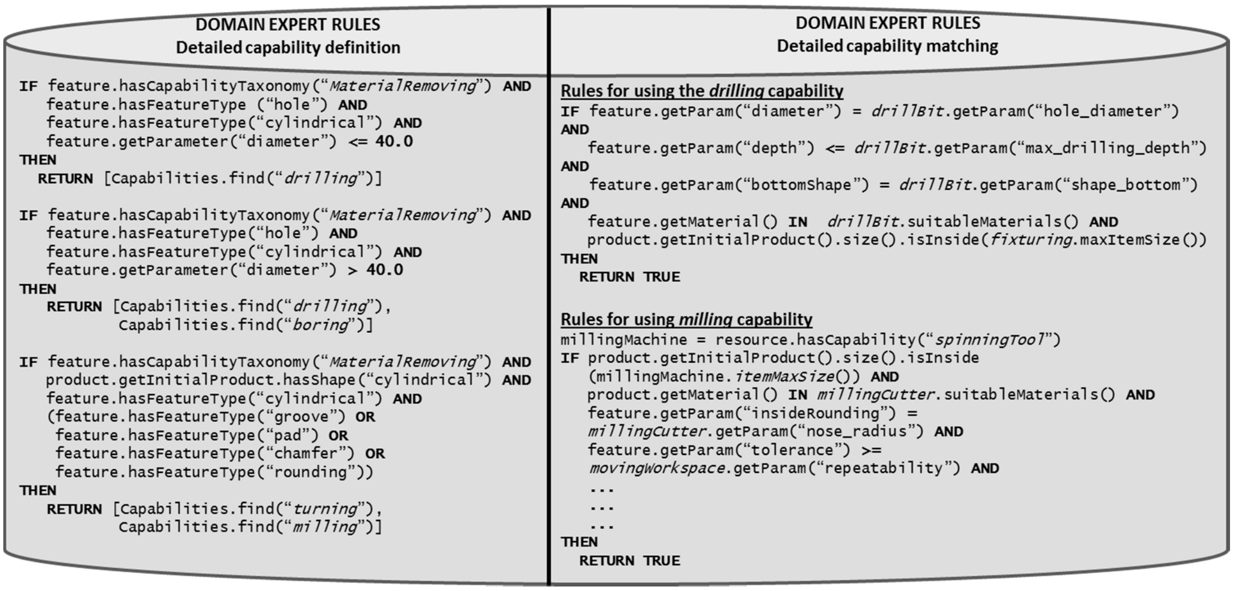
As stated in the previous chapter, the detailed capability matching, which considers the capability parameters, is based on rule-based reasoning. For this, three types of rules are needed, comprising:

1) Rules for defining how the capability and product feature information are applied in different domains when matching the product requirements with the resource capabilities;
2) Rules for reasoning out the parameters of the combined capabilities;
3) Rules indicating how adaptation guidelines and other criteria, such as availability and scheduling, device condition and lifecycle, and user and company-specific criteria, are used in the final selection of resources and generating the final configuration.

These domain expert rules, combined capability rules and adaptation rules, as well as their use in the capability matching, will be discussed in this chapter, and a few examples of the rules will be given in pseudo code. The example rules have been constructed to serve the demonstrations in the TUT-machining laboratory and the TUT-microfactory environments, discussed in the case study chapter (Chapter 5). The rules have been defined to cover these (and similar) use cases. This chapter presents only some examples of the rules, while the rest can be found in Appendix 5.

Domain expert rules

Domain expert rules include rules which define how the product feature and resource capability information are applied in different domains when matching the product requirements with suitable resources. The domain expert rules are divided into two categories. First, there are detailed capability definition rules for defining which capability should be used to fulfil a certain type of product requirement from the pre-process plan. The second types of rules are detailed capability matching rules for defining whether the available capabilities are suitable for the order at the capability parameter level. These latter rules take into account, e.g. in the machining process domain, how the achieved feature depends on the tool shape and type (e.g. in the milling process, what kind of roundings and chamfers can be achieved with certain types and shapes of tools). By using these domain expert rules it is possible, for example, to first detect that the "milling" capability should be used, and then to compare whether the available "milling" capabilities have the parameters needed in order to produce the required feature. One domain expert rule is, for example, "In milling, the nose radius of the cutter has to be same as the required rounding inside the machined pocket". More examples of the domain expert rules written in pseudo code can be found in Figure 35. The defined rules are definitely not comprehensive, but can be extended gradually. The focus here is not on defining all the possible domain-specific rules in detail, but to demonstrate the concept of how to use these rules and capability definitions in the design and adaptation of a production system.



DOMAIN EXPERT RULES
Detailed capability definition

```
IF feature.hasCapabilityTaxonomy("MaterialRemoving") AND
   feature.hasFeatureType ("hole") AND
   feature.hasFeatureType("cylindrical") AND
   feature.getParameter("diameter") <= 40.0
THEN
  RETURN [Capabilities.find("drilling")]

IF feature.hasCapabilityTaxonomy("MaterialRemoving") AND
   feature.hasFeatureType("hole") AND
   feature.hasFeatureType("cylindrical") AND
   feature.getParameter("diameter") > 40.0
THEN
   RETURN [Capabilities.find("drilling"),
           Capabilities.find("boring")]

IF feature.hasCapabilityTaxonomy("MaterialRemoving") AND
   product.getInitialProduct.hasShape("cylindrical") AND
   feature.hasFeatureType("cylindrical") AND
   (feature.hasFeatureType("groove") OR
    feature.hasFeatureType("pad") OR
    feature.hasFeatureType("chamfer") OR
    feature.hasFeatureType("rounding"))
THEN
   RETURN [Capabilities.find("turning"),
           Capabilities.find("milling")]
```

DOMAIN EXPERT RULES
Detailed capability matching

Rules for using the *drilling* capability

```
IF feature.getParam("diameter") = drillBit.getParam("hole_diameter")
AND
   feature.getParam("depth") <= drillBit.getParam("max_drilling_depth")
AND
   feature.getParam("bottomShape") = drillBit.getParam("shape_bottom")
AND
   feature.getMaterial() IN  drillBit.suitableMaterials() AND
   product.getInitialProduct().size().isInside(fixturing.maxItemSize())
THEN
  RETURN TRUE
```

Rules for using *milling* capability

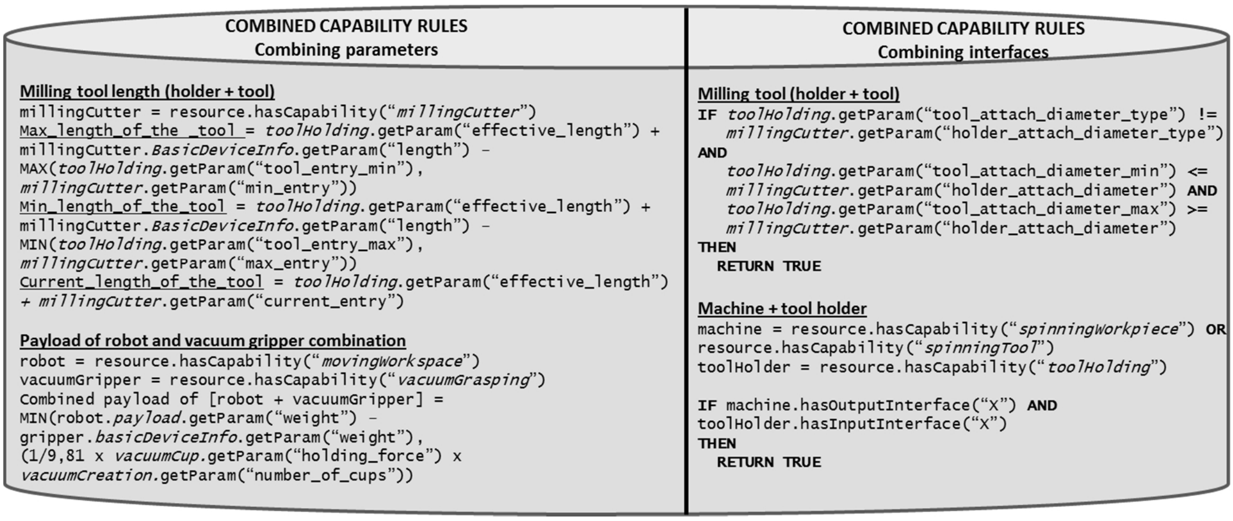
```
millingMachine = resource.hasCapability("spinningTool")
IF product.getInitialProduct().size().isInside
   (millingMachine.itemMaxSize()) AND
   product.getMaterial() IN millingCutter.suitableMaterials() AND
   feature.getParam("insideRounding") =
   millingCutter.getParam("nose_radius") AND
   feature.getParam("tolerance") >=
   movingWorkspace.getParam("repeatability") AND
   ...
   ...
   ...
THEN
  RETURN TRUE
```

Figure 35 . Examples of the domain expert rules. The capability concept names are written in italics.

In some rules, not only in domain expert rules, but also in others, some variable names are defined at the beginning of the rule. This is done in order to be able to separate different devices which have the same capabilities, but different roles in the device combinations. For example "itemMaxSize" is a common capability used to indicate the maximum item size that the device can handle and it can relate to almost any functional capability. In the device combination under analysis, there may be multiple devices having this same capability. Defining the variable name at the beginning of the rule allows the capabilities of the correct device to be referred to.

Combined capability rules

Combined capability rules are needed when combining resources and analyzing their emergent capabilities. A combined capability is not simply the sum of the individual capabilities it is composed of. These rules are also divided into two groups: combining parameters and combining interfaces rules (see Figure 36). The former determine how the parameters of the combined capabilities can be formed from the parameters of the individual capabilities, whereas the latter are meant to check whether the interfaces between the devices to be combined are compatible. One rule is, for example, "When a robot and gripper are combined, the payload of the combination is the robot payload minus the gripper weight or gripper payload, provided this is smaller than the previous difference". Other rules relate, for example, to accuracy and the workspace of the combination of devices. The aim of the combined capability rules is not to provide detailed analysis of, e.g. the workspace or the kinematics of the device, but to enable the modelling of scenarios for potentially suitable device combinations. For kinematics and detailed workspace definitions, virtual simulation tools should be used to validate the results obtained from the reasoning based on the digital information.



COMBINED CAPABILITY RULES
Combining parameters

Milling tool length (holder + tool)

```
millingCutter = resource.hasCapability("millingCutter")
Max_length_of_the _tool = toolHolding.getParam("effective_length") +
millingCutter.BasicDeviceInfo.getParam("length") –
MAX(toolHolding.getParam("tool_entry_min"),
millingCutter.getParam("min_entry"))
Min_length_of_the_tool = toolHolding.getParam("effective_length") +
millingCutter.BasicDeviceInfo.getParam("length") –
MIN(toolHolding.getParam("tool_entry_max"),
millingCutter.getParam("max_entry"))
Current_length_of_the_tool = toolHolding.getParam("effective_length")
+ millingCutter.getParam("current_entry")
```

Payload of robot and vacuum gripper combination

```
robot = resource.hasCapability("movingWorkspace")
vacuumGripper = resource.hasCapability("vacuumGrasping")
Combined payload of [robot + vacuumGripper] =
MIN(robot.payload.getParam("weight") –
gripper.basicDeviceInfo.getParam("weight"),
(1/9,81 x vacuumCup.getParam("holding_force") x
vacuumCreation.getParam("number_of_cups"))
```

COMBINED CAPABILITY RULES
Combining interfaces

Milling tool (holder + tool)

```
IF toolHolding.getParam("tool_attach_diameter_type") !=
   millingCutter.getParam("holder_attach_diameter_type")
AND
   toolHolding.getParam("tool_attach_diameter_min") <=
   millingCutter.getParam("holder_attach_diameter") AND
   toolHolding.getParam("tool_attach_diameter_max") >=
   millingCutter.getParam("holder_attach_diameter")
THEN
  RETURN TRUE
```

Machine + tool holder

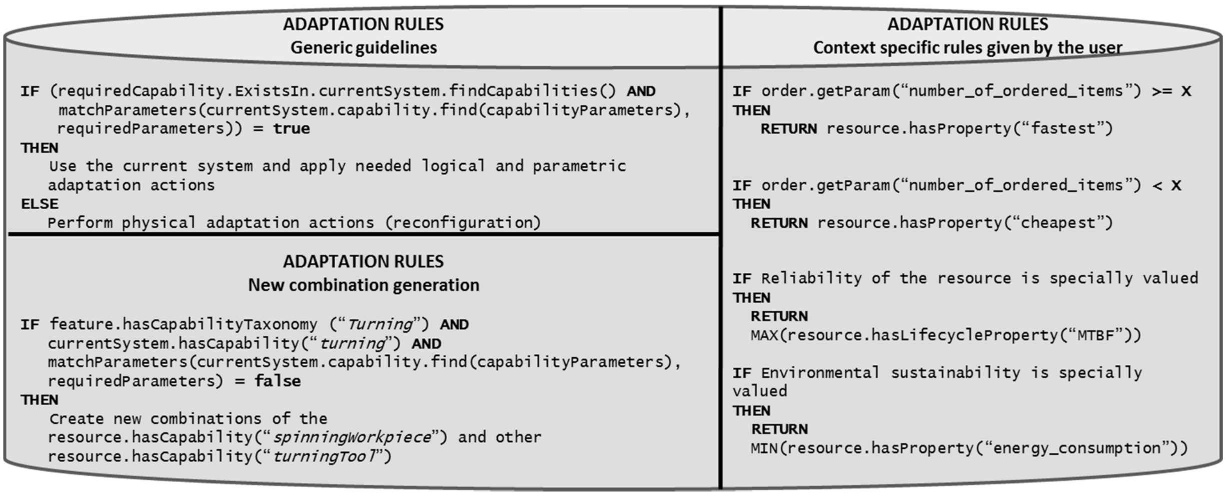
```
machine = resource.hasCapability("spinningWorkpiece") OR
resource.hasCapability("spinningTool")
toolHolder = resource.hasCapability("toolHolding")

IF machine.hasOutputInterface("X") AND
toolHolder.hasInputInterface("X")
THEN
  RETURN TRUE
```

Figure 36. Examples of the combined capability rules.

It has to be noted that in many cases the combined capability rules developed here produce highly simplified and crude results. This is due to the problem that many of the properties of the combined capabilities emerge as a behavior of the machine as a whole in a certain context, and they cannot be decomposed into the properties of the various components. For example, reasoning out the combined accuracy of a machining center and its attached tool is a complex problem, because it is affected by multiple factors other than those included in the resource description. For instance, the accuracy and repeatability of machine tools are limited by geometric errors of all the components, kinematic errors, load-induced errors, thermal errors, dynamic errors, calibration errors and computational errors. The geometric accuracy of a machine tool is determined by the errors that exist in the machine tool from its basic design, the inaccuracies built-in during assembly and from the components used in the machine. There are other sources of error too, such as the cutting force, the humidity, the competence of the operator, tool wear and so on. An experienced operator is usually able to compensate for most of these errors. Therefore, in the case of machining operations, the repeatability of the machine is usually a more important parameter than its accuracy. As this example indicates, some properties cannot be traced back to any specific property of the individual components but emerges in a very complex way. For determining this, a comprehensive model of the whole machine, its control and the manufacturing process should be constructed, which is beyond the scope of this work. Another difficulty in defining the accuracy of a machine tool is that usually neither the tool holder nor the tool tolerances, are available in the catalogues.

Adaptation and configuration rules

The third type of rules no longer relate directly to capability matching, but to what happens after the devices possessing the adequate capabilities have been found, or if no match has been found. The adaptation rules include rules defining how other criteria, such as resource availability and scheduling information, device condition and lifecycle, as well as user- and company-specific criteria relating to, for example, costs or eco-efficiency, are used in the final resource selection and generation of the configuration. There are three types of defined adaptation rules. The generic guidelines are used to define the type of adaptation needed when the requirement changes (physical, logical or parametric), whereas the rules for generating a new combination define the principles for generating new resource combinations if the existing ones don't provide the required capabilities. If multiple resources or resource combinations provide the required capabilities, other criteria should be applied to select a suitable configuration for the given situation. The context-specific rules have been defined for such situations. These are usually given by the user in a specific case and are therefore dynamic in their nature. One user-defined adaptation rule could be, for example, “If the amount of ordered items is 20 or more, use the fastest machine for manufacturing the product. Otherwise, use the cheapest option.” More examples of adaptation rules are shown in Figure 37.



ADAPTATION RULES
Generic guidelines

```
IF (requiredCapability.ExistsIn.currentSystem.findCapabilities() AND
    matchParameters(currentSystem.capability.find(capabilityParameters),
    requiredParameters)) = true
THEN
   Use the current system and apply needed logical and parametric
   adaptation actions
ELSE
   Perform physical adaptation actions (reconfiguration)
```

ADAPTATION RULES
New combination generation

```
IF feature.hasCapabilityTaxonomy (“Turning”) AND
   currentSystem.hasCapability(“turning”) AND
   matchParameters(currentSystem.capability.find(capabilityParameters),
   requiredParameters) = false
THEN
   Create new combinations of the
   resource.hasCapability(“spinningWorkpiece”) and other
   resource.hasCapability(“turningTool”)
```

ADAPTATION RULES
Context specific rules given by the user

```
IF order.getParam(“number_of_ordered_items”) >= X
THEN
   RETURN resource.hasProperty(“fastest”)


IF order.getParam(“number_of_ordered_items”) < X
THEN
  RETURN resource.hasProperty(“cheapest”)


IF Reliability of the resource is specially valued
THEN
  RETURN
  MAX(resource.hasLifecycleProperty(“MTBF”))

IF Environmental sustainability is specially
valued
THEN
  RETURN
  MIN(resource.hasProperty(“energy_consumption”))
```

Figure 37. Examples of the adaptation and configuration rules.

The adaptation relies strongly on the context-specific information. As the computerized management of the context-specific information is still not mature enough, the adaptation rules are mainly used by humans as tools for reasoning. The demonstration environment constructed here is a human-machine environment, which assumes a high degree of human involvement. Particularly in the adaptation phase, human intelligence is required to perform the context-specific reasoning, such as being able to recognize the cheapest production resource for a given order. In order to reliably ascertain which resource is the cheapest in a specific context, multiple parameters need to be considered (e.g. energy consumption, resource and consumable consumption, the human resources needed, the maintenance actions required and so on). Then, optimization algorithms based on those parameters have to be formulated. Naturally, an in-depth process knowledge is also required for this purpose, and the development of such an algorithm is beyond the scope of this work.

4.4.5. Detailed capability matching

The detailed capability matching is done based on the rules included in the rule-base introduced in the previous chapter. This chapter is devoted to explaining how the rules are applied during the detailed capability matching. The starting point here is the pre-process plan, in which only the generic capability requirements such as “MaterialRemoving” are defined. Figure 38 represents the framework for capability matching in the developed ICT-architecture. This ICT-architecture will be discussed in Chapter 5.1.

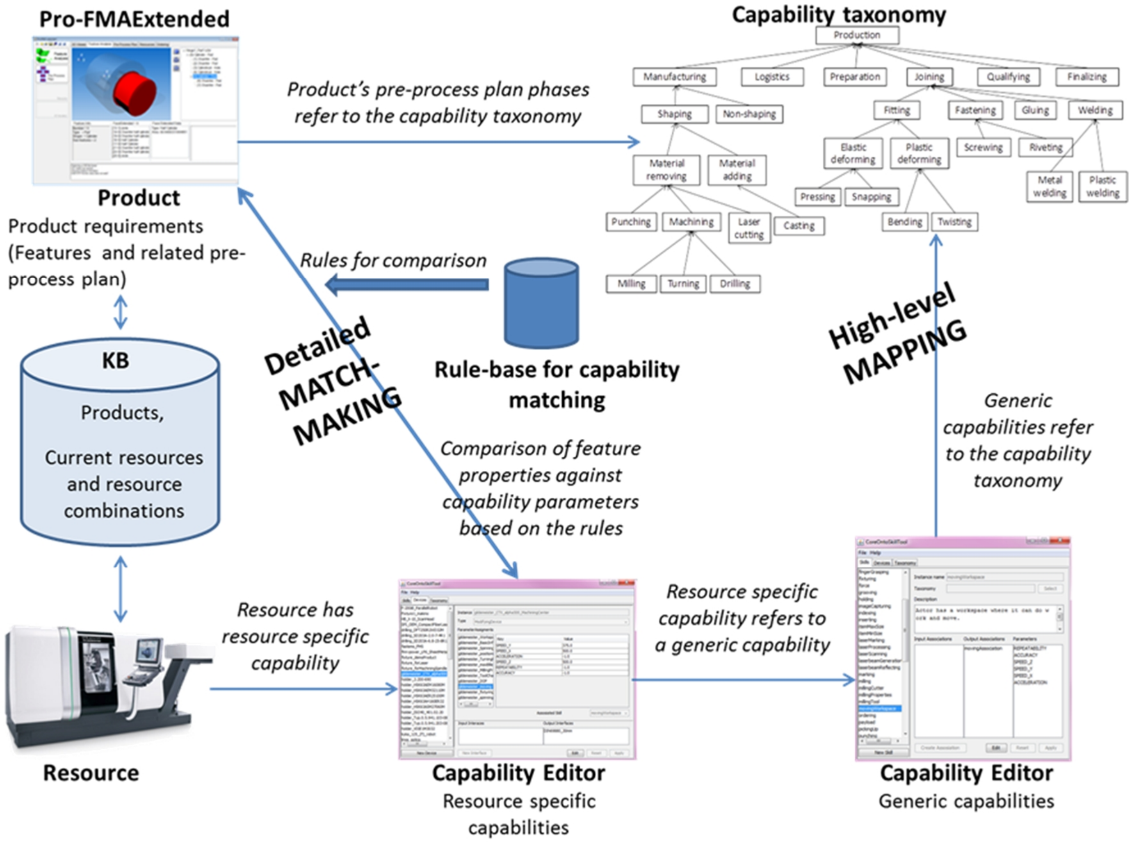


Figure 38. Matching the product requirements against the system capabilities (Järvenpää et al. 2012b).

The task of interpreting and applying the rules belongs to the rule inference engine. According to Hopgood (2001) the application of rules can be broken down as follows: 1) selecting rules to examine — these are all the available rules; 2) determining which of these are applicable — these are the triggered rules; 3) selecting which rule to execute.

In order to limit the number of rules that have to be triggered in each capability matching case, the rules are first classified according to their purpose and then they are linked to the capability taxonomy level to which they relate, see Figure 39. This means that if the product requirement relates to the "milling" capability, the rule engine will trigger only those rules that are linked to "Milling" in the capability taxonomy. In Figure 39, the white boxes with dashed outlines indicate the taxonomy levels to which the rules are linked. The figure gives a simplified representation, meaning that only a limited number of examples of rule types are shown. The generic guidelines and context-specific rules for adaptation are not capability related, and do not therefore have any links to the taxonomy.

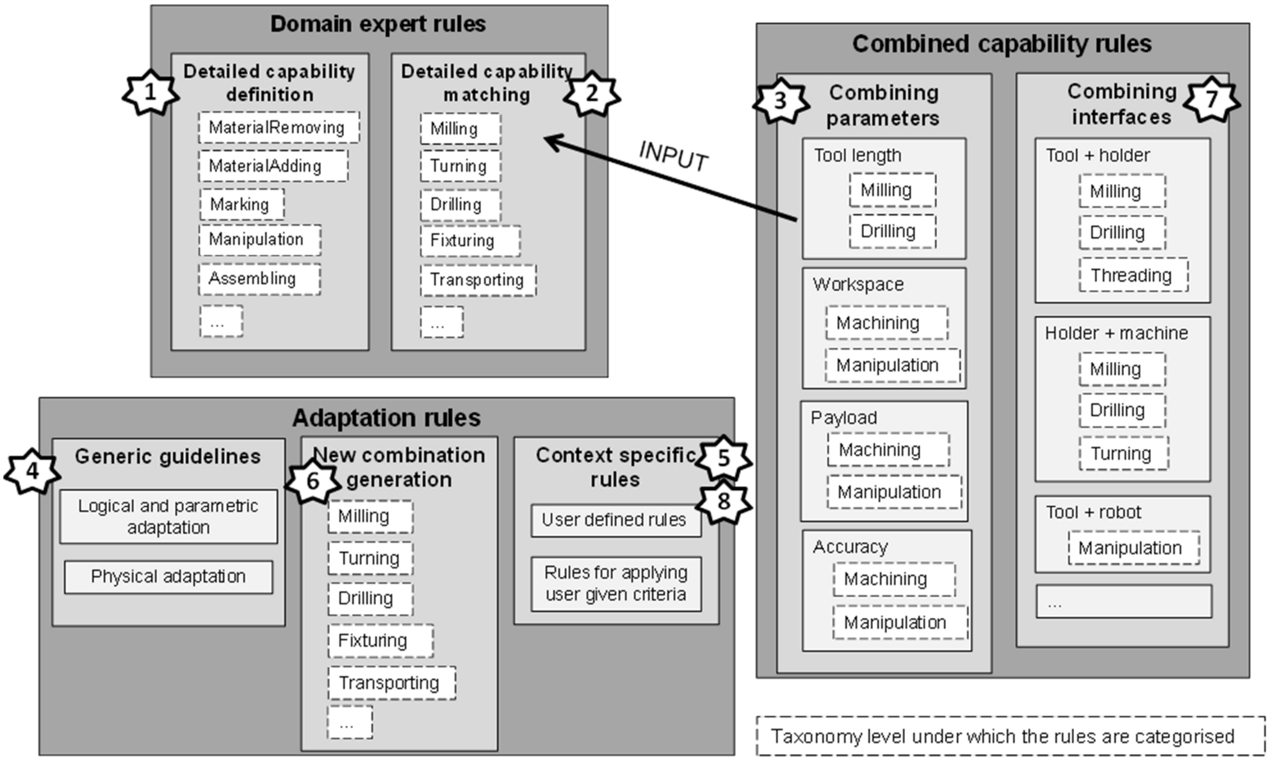


Figure 39. Categorisation of the rules (Järvenpää et al. 2012b).

The numbering in Figure 39 indicates the sequence in which the rules are applied when a new order enters the system. The order is sent in the form of a pre-process plan, which indicates the features to be manufactured and their link to the capability taxonomy. The pre-process plan may define, for example, that some sort of “material removing” capability is required. When an order is sent to the system, the capability matching proceeds as follows by utilizing the rules:

1. The detailed capability is first defined based on the given taxonomy level, feature information and the domain expert rules for detailed capability definition. If the pre-process plan referred to “MaterialRemoving” in the capability taxonomy, only the domain expert rules linked with “MaterialRemoving” will be triggered. The rules are checked one by one until a match is found. The reasoning may conclude that, for example, a “*drilling”* capability is required.

2. After the detailed capability requirements have been defined, and the capabilities of the existing system which match the capability requirements at the concept name level have been identified, these provided and requested capabilities need to be compared at the parameter level. This happens with the domain expert rules for detailed capability matching.

3. The detailed capability matching (phase 2) often needs input from phase three, where the parameters of the individual device capabilities are combined to derive the parameters of the combined capabilities. These combined capability rules for combining the parameters are implemented, for example, to reason out the workspace, payload or accuracy of a resource combination. Different rules belonging to the same rule category may also need input from each other. For example, in order to evaluate the workspace, the tool length

needs to be known. These rules are triggered when called for by the detailed capability matching rules or other combining parameters rules.

4. Based on the matching results from phases 2 and 3 and the generic guidelines in the adaptation rules category, the need for logical and parametric adaptation or physical adaptation will be defined.

5. If all the required capabilities were found in the current system, no physical adaptation is required and multiple alternative resources can satisfy the order, so context specific adaptation rules are applied to select the most suitable resource configuration for the given order.

6. If all the required capabilities were not available in the current system, physical adaptation actions need to be performed. This means that new combinations of resources need to be generated. For this, the new combination generation rules will be applied. These rules aim to utilize the existing devices on the factory floor (e.g. a lathe) and combine them with new resources (e.g. cutting tools) in order to change the system's capability to match the given order. If the existing devices on the factory floor can not provide suitable capabilities when combined with other devices, completely new combinations, built from devices not currently existing on the factory floor, will be created. These rules are also linked to the capability taxonomy, and depending on which new capability is needed, only the relevant rules will be triggered.

7. In order to form new combinations of devices having simple capabilities which could form the required combined capabilities, the interface compatibility of the devices needs to be ensured. The combined capability rules for combining the interfaces will be utilized to achieve this. After this, the detailed match of the new device combination to the requirements will be checked, as was done in phase 2.

8. Finally, if more than one new combination of suitable devices was found, context specific adaptation rules will be used in order to select the most desirable combination for the given situation.

The presented rules and associated matching procedures provide a valuable aid for selecting appropriate solutions from a large amount of input data. For example, a large factory with thousands of resources represents a huge solution space, for which manual handling in the context of such capability matching is cumbersome. With the approach presented here, it is possible to explore the large solution space and rapidly filter out the unsuitable resources, leaving only the possible resources and resource combinations for the given requirement. Therefore it eases and speeds up the adaptation planning, and facilitates the reactive adaptation.

4.5. Information formalization to support production system adaptation - Implementation

In the proposed approach, ontological modelling is used to formalize the representation of the resources, capabilities and system configurations. The Core Ontology defined by Lanz (2010) is used as a basis for describing the product-, process- and system-related information. The knowledge representation can be interpreted by both humans and machines. The ontology has been built with the Protégé OWL-DL tool. OWL-DL is based on description logics and it allows the domain concepts to be largely defined according to a predefined formalism. (Stanford Center for Biomedical Informatics Research 2012.)

For this study, the Core Ontology has been updated and extended to better answer the problem at hand, in other words to better fulfil the requirements set by the adaptation of production systems in a changing environment. A new domain, capability, has been defined to connect the process and system domains. Capabilities can be seen as an intermediate step between systems and processes. Systems utilize their capabilities to execute processes. The following chapters first present the formalization of the capability and production system information in the Core Ontology. After this, the formalization of the connection between the product, process and system information is discussed. The work done during the course of this thesis has also slightly affected the product and process domains in the Core Ontology, but these are not discussed here. Finally the chapter ends with a discussion of the spatial and temporal reasoning enabled by the ontology definitions.

4.5.1. The formalization of the capabilities in the Core Ontology

This chapter will explain the structure of the capability definition in the Core Ontology, its sub-classes, their parameters and the relationships between the classes. The CapabilityDefinition class, displayed in Figure 40, contains the sub-classes needed for managing the simple and combined capabilities of resources. As the theory and concepts behind the capability modelling has already been thoroughly discussed in the previous chapters, the theory needs no further explanation here. Instead, the emphasis is placed on presenting the structure of the knowledge formalization.

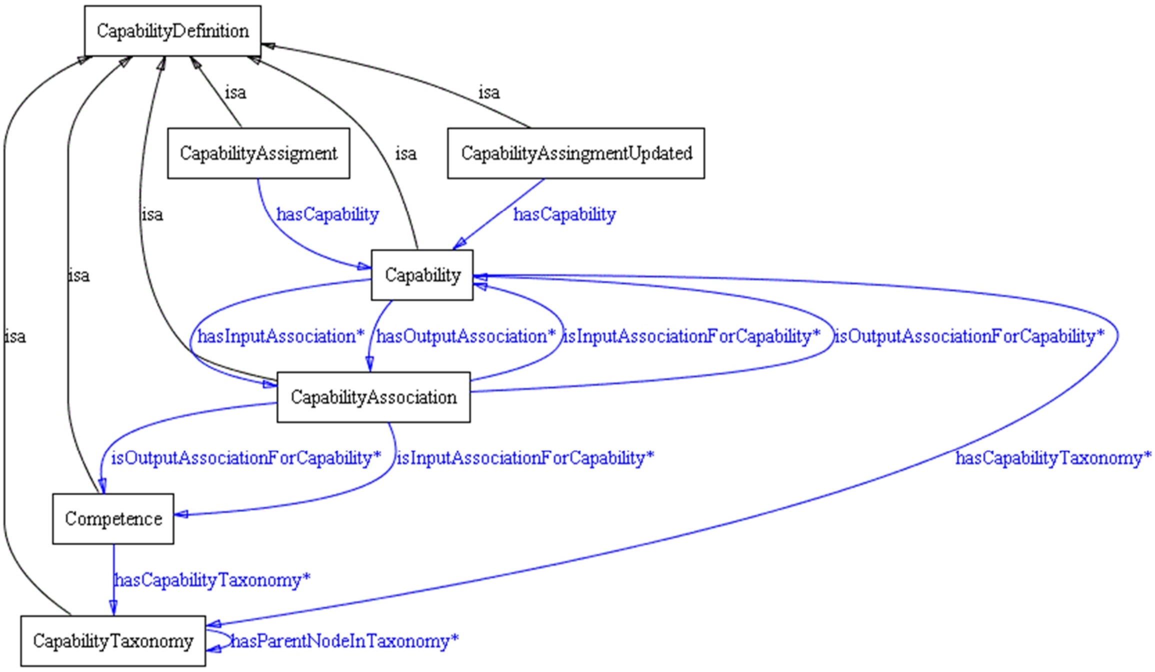


Figure 40. Class CapabilityDefinition and its sub-classes.

Capability

The capability class defines the pool of generic capabilities that can exist in a production system, including the simple and combined capabilities. Both the strong and weak capabilities are defined in this class. In the current implementation, the business and lifecycle properties are also given as capabilities. The names of the capability instances represent the capability concept name as discussed in Chapter 4.2.1. Table 11 explains the relevant properties of the instances belonging to the Capability class. A few default properties, namely *description* (giving the description of the entity) and *name* (defining the name of the entity), have been excluded from the tables in order to save space.

Table 11. Properties of the Capability class.

Property name	Description
capabilityParameter	CapabilityParameter contains a list of parameters that a certain capability can have. The parameter values are defined when the capability is assigned to a resource.
capabilityType	This property is used to indicate if the defined capability is a normal capability or if it is used to describe the business or lifecycle properties.
hasCapabilityTaxonomy	This property links the capabilities with the capability taxonomy, allowing the mapping of product requirements against resource capabilities. The reference to the CapabilityTaxonomy is only defined for the capability instances representing strong capabilities.
hasInputAssociation	This property indicates which capability associations are needed as an input in order to enable the current capability to emerge (e.g. moving and holding associations are needed as an input for transporting).
hasOutputAssociation	This property indicates which output associations the current capability provides for the combined capabilities as their input (e.g. moving provides input for transporting).

CapabilityAssociation

The CapabilityAssociation class contains the association instances which are used to create links between capabilities, enabling the formulation of combined capabilities from simple capabilities. The associations can be either input or output associations. For example, simple capabilities provide output associations for combined capabilities as their input association, and vice versa. Table 12 shows the properties of the CapabilityAssociation class. These properties are the inverse of *hasOutputAssociation* and *hasInputAssociation* properties. They help the manual management of the capability information in the Protégé.

Table 12. Properties of the CapabilityAssociation class.

Property name	Description
isInputAssociationForCapability	This property indicates for which combined capability this is an input.
isOutputAssociationForCapability	This property indicates for which capability this is an output.

CapabilityTaxonomy

The CapabilityTaxonomy class contains the hierarchical categorization of the possible functional capabilities. Capabilities and Processes refer to certain levels in the CapabilityTaxonomy enabling the product requirements to be mapped with the device capabilities at different levels of detail. Table 13 shows the properties of the CapabilityTaxonomy class.

Table 13. Properties of the CapabilityTaxonomy class.

Property name	Description
hasParentNodeInTaxonomy	The hasParentNodeInTaxonomy property defines the parent node for the current taxonomy node. It is used to create the hierarchy between different detail-level capabilities (functions vs. behaviours) and therefore to build the taxonomy, e.g. milling hasParentNodeInTaxonomy machining.

Competence

The Competence class is meant for storing the capabilities of a human or organization. This class is under construction and beyond the scope of this work.

CapabilityAssignment

The CapabilityAssignment class defines the resource-specific capabilities and their parameter values. CapabilityAssignments are individual instantiations of the generic capabilities in the Capability class defining the resource-specific capabilities. Table 14 shows the properties of the CapabilityAssignment class.

Table 14. Properties of the CapabilityAssignment class.

Property name	Description
hasCapability	This property refers to the generic capability that this CapabilityAssignment is an instantiation of.
hasParams	This property allows the assignment of parameters for the entity. It refers to key - value pairs.

CapabilityAssignmentUpdated

The CapabilityAssignmentUpdated class is similar to the CapabilityAssignment class, but it is meant for storing the updated capability information during the individual device's lifecycle. It has the same properties as the CapabilityAssignment class shown in Table 14. The lifecycle properties can be assigned through this class.

4.5.2. Formalization of resources in the Core Ontology

This chapter will explain the structure of the resource definitions in the Core Ontology, its sub-classes, their parameters and the relationships between the classes. Figure 41 shows the ontology structure for representing the resource information. Those sub-classes of the ontology which are not used or developed in the context of this thesis, will not be discussed here. For more information about them, please refer to Lanz (2010).

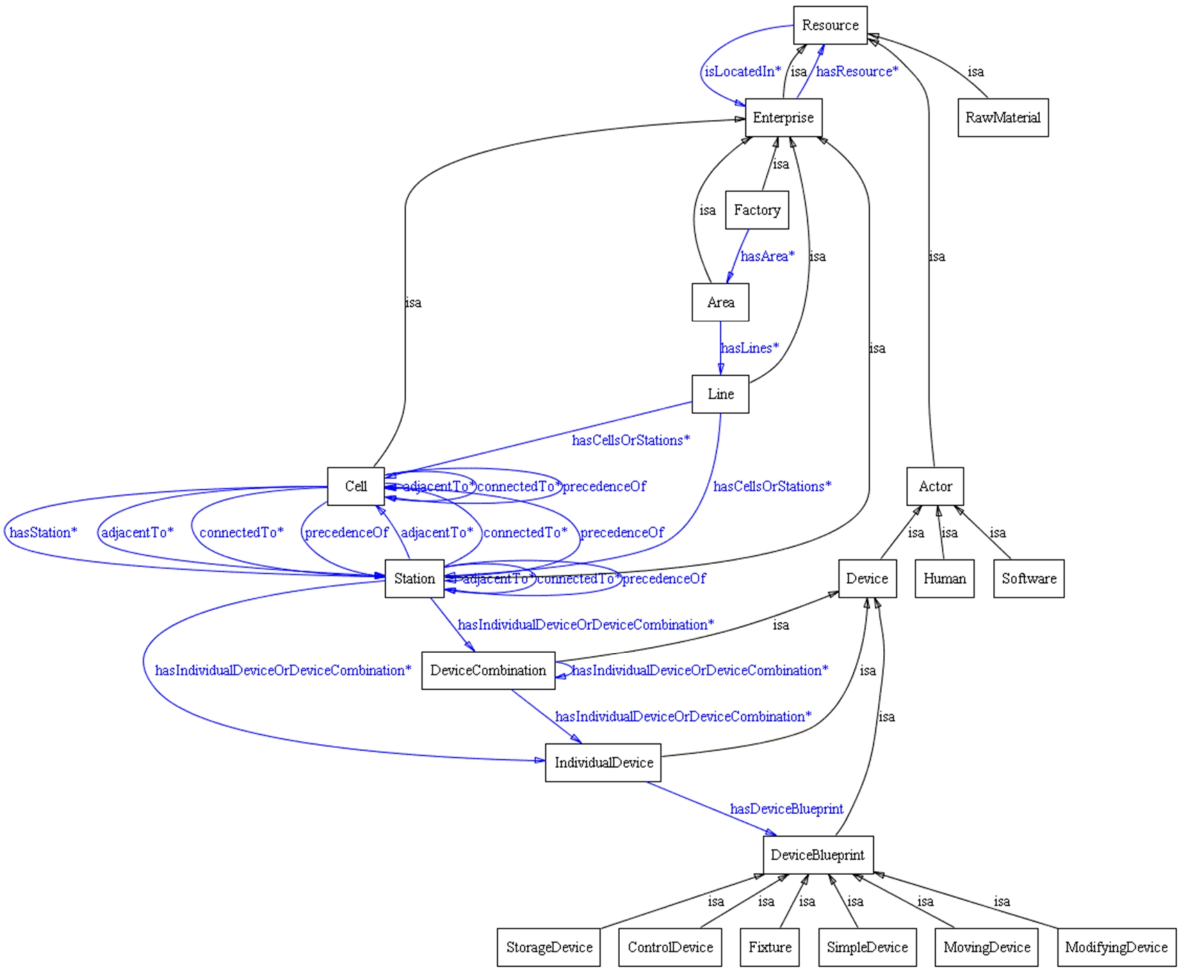


Figure 41. Class Resource and its sub-classes.

Resource

The class Resource refers to any physical or virtual resource, such as a human, a device or software, or a facility area that is used for manufacturing or assembling a product element or providing a certain service. Table 15 shows the properties of the Resource class.

Table 15. Properties of the Resource class

Property name	Description
IMGModelUrl	IMGModelUrl defines the URL for the picture (e.g. .png) of the given object.
isLocatedIn	This property indicates the place where the entity is located.

Device

The Device class includes any machines, equipment and tools used to manufacture or assemble a product element. Devices can be either catalogue devices (blue prints) or actual individual devices and combinations of them. Figure 42 shows the class Device and its sub-classes and Table 16 explains its properties.

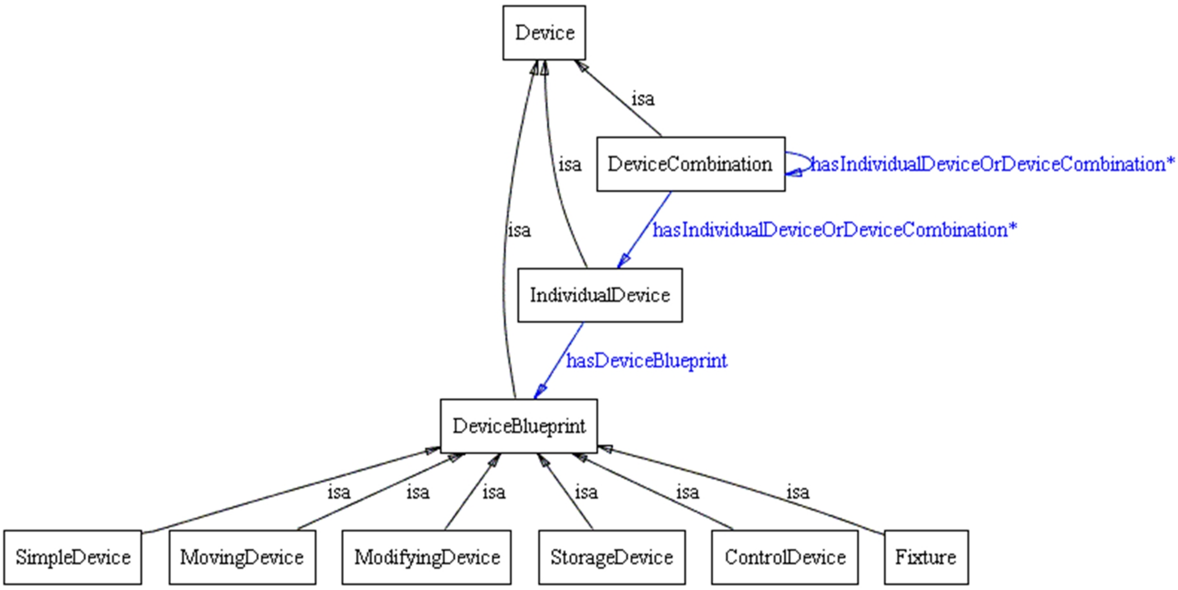


Figure 42. Class Device and its sub-classes.

Table 16. Properties of Device class.

Property name	Description
deviceID	The deviceID is a unique identifier of a device.
IMGModelUrl	IMGModelUrl defines the URL for the picture (e.g. .png) of the given object.
isLocatedIn	This property indicates the place where the entity is located. It can be, e.g. a certain cell or workstation.

DeviceBlueprint

The DeviceBlueprint class contains the catalogue information of different device types. The DeviceBlueprint has the subclasses ControlDevice, Fixture, ModifyingDevice, MovingDevice, StorageDevice and SimpleDevice, as defined by Lanz (2010), which all inherit the parameters of the DeviceBlueprint class. Basically the division is only meant for humans and it helps to manage the device information in the ontology. Table 17 shows the properties of the DeviceBlueprint class.

Table 17. Properties of the DeviceBlueprint class.

Property name	Description
deviceID	The deviceID is a unique identifier of a device.
IMGModelUrl	IMGModelUrl defines the URL for the picture (e.g. .png) of the given object.
isLocatedIn	This property indicates the place where the entity is located. As the DeviceBlueprint instances are not real physical resources, they should only be located in catalogues. Currently, the ontology doesn’t restrict this.
outputInterface	The output interface is the interface the device provides for other devices to connect to, e.g. a tool holder provides an output interface for a tool.
inputInterface	The Input interface is the interface the device use to connect to other devices, e.g. a tool holder has an input interface, which is used to connect it to the machine.
hasCapabilityAssigment	This property links the resources and their capabilities.

IndividualDevice

The InvidualDevice class is meant for saving the instances of the actual individual devices existing on the factory floor. The individual devices have reference to the DeviceBlueprint, based on which their nominal capability can be known. The updated capability information during the individual device lifecycle, i.e. the actual capability, is assigned to these instances by the *hasCapabilityAssignmentUpdated* property. Table 18 shows the properties of the IndividualDevice class.

Table 18. Properties of the IndividualDevice class.

Property name	Description
deviceID	The DeviceID is a unique identifier of a device.
IMGModelUrl	IMGModelUrl defines the URL for the picture (e.g. .png) of the given object.
hasDeviceBlueprint	HasDeviceBlueprint links the individual device to the blueprint (i.e. to the catalogue information).
isLocatedIn	This property indicates the place where the entity is located, e.g. in a certain cell or station.
hasPosition	This property defines the position of the entity.
hasOrientation	This property defines the orientation of the entity.
hasCapabilityAssigmentUpdated	This property links the resources and their updated capabilities during the individual device's lifecycle.

DeviceCombination

The DeviceCombination class includes combinations of multiple individual devices or other device combinations. Table 19 shows the properties of the DeviceCombination class.

Table 19. Properties of the DeviceCombination class.

Parameter name	Description
deviceID	The DeviceID is a unique identifier of a device.
IMGModelUrl	IMGModelUrl defines the URL for the picture (e.g. .png) of the given object.
hasIndividualDeviceOrDeviceCombination	This property defines the individual devices and other device combinations that are included in the current device combination.
isLocatedIn	This property indicates the place where the entity is located, e.g. a certain cell or station.

Enterprise

The Enterprise class refers to the company or facility where the operations take place. It is a parent class for other facility-related classes. It answers the question "where". The Enterprise class structure is directly adopted from the work of Lanz (2010), but the properties are slightly modified. The relations between the cells and stations have been added in this work. Figure 43 shows the structure of the Enterprise class and its sub-classes, whereas in Table 20 the properties of the Enterprise class are explained. This clearly illustrates the restrictions of OWL, which doesn't allow the representation of the *is_part_of* relation. Therefore, the lines, cells and stations are modelled with a relation *is_a* enterprise, even though logically they should be *part_of*.

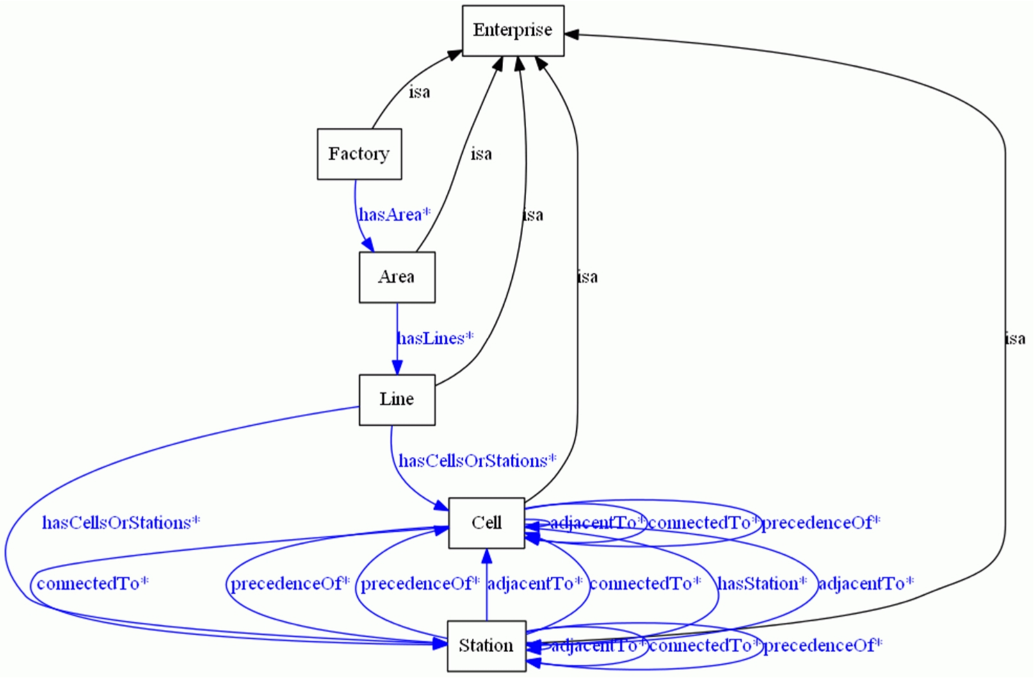


Figure 43. Class Enterprise.

It is intended in the future to break up this hierarchical line-cell-station structure, because it is highly dependent on the company's or user's naming policies. Some may call a cell a station, and vice versa. This representation restricts the allowable system architecture. For the purpose of the adaptation problem, it would be enough to be able to present the combinations of resources and their relative locations in the factory layout.

Table 20. Properties of the Enterprise class.

Property name	Description
IMGModelUrl	IMGModelUrl defines the URL for the picture (e.g. .png) of the given object.
isLocatedIn	This property indicates the place where the entity is located.
hasResource	The hasResource property defines the resources that belong to the given entity.

Factory

The class Factory refers to a complete factory where the manufacturing or assembly takes place. Its properties are shown in Table 21.

Table 21. Properties of the Factory class.

Property name	Description
IMGModelUrl	IMGModelUrl defines the URL for the picture (e.g. .png) of the given object.
isLocatedIn	This property indicates the place where the entity is located.
hasResource	The hasResource property defines the resources that belong to the given entity.
hasArea	This property defines the areas that are located in the factory.

Area

The class Area refers to a place where the assembly or manufacturing takes place. Its properties are shown in Table 22.

Table 22. Properties of the Area class.

Property name	Description
IMGModelUrl	IMGModelUrl defines the URL for the picture (e.g. .png) of the given object.
isLocatedIn	This property indicates the place where the entity is located.
hasResource	The hasResource property defines the resources that belong to the given entity.
hasLines	This property indicates the lines that are located in the area.

Line

The class Line refers to the production line where the manufacturing or assembly takes place. A line can consist of multiple cells and/or stations. Table 23 shows the properties of the class Line.

Table 23. Properties of the Line class.

Property name	Description
IMGModelUrl	IMGModelUrl defines the URL for the picture (e.g. .png) of the given object.
isLocatedIn	This property indicates the place where the entity is located.
hasResource	The hasResource property defines the resources that belong to the given entity.
hasCellsOrStations	This property defines the cells or stations that are included in the line.
hasOrientation	This property defines the orientation of the entity.
hasPosition	This property defines the position of the entity.

Cell

The class Cell refers to the cell where the manufacturing or assembly takes place. A cell can be either a product-oriented or process-oriented cell. A process-oriented cell may consist of multiple machines providing similar process capabilities. In contrast, a product-oriented cell consists of resources targeted to manufacture certain entities of the product (e.g. sub-assemblies). The properties of the class Cell are shown in Table 24.

Table 24. Properties of the Cell class.

Property name	Description
IMGModelUrl	IMGModelUrl defines the URL for the picture (e.g. .png) of the given object.
isLocatedIn	This property indicates the place where the entity is located.
hasResource	The hasResource property defines the resources that belong to the given entity.
adjacentTo	This property indicates which entities are adjacent to (close to, next to) the current entity, but not physically connected.
connectedTo	This property indicates the entities which are physically connected to the current entity.
precedenceOf	This property defines which entity precedes the current entity in the layout.
hasStation	This property indicates which stations include the given cell.
hasPosition	This property defines the position of the entity.
hasOrientation	This property defines the orientation of the entity.

Station

The Station class refers to a station where the manufacturing or assembly takes place. A station refers to one unit in the system, where certain operations can be completed. A cell can contain multiple stations. The properties of the class Station are shown in Table 25.

Table 25. Properties of the Station class.

Property name	Description
IMGModelUrl	IMGModelUrl defines the URL for the picture (e.g. .png) of the given object.
isLocatedIn	This property indicates the place where the entity is located.
hasResource	The hasResource property defines the resources that belong to the given entity.
adjacentTo	This property indicates which entities are adjacent to (close to, next to) the current entity, but not physically connected.
connectedTo	This property indicates the entities which are physically connected to the current entity.
precededBy	This property defines which entity precedes the current entity in the layout.
hasIndividualDeviceOrDeviceCombination	This property defines the individual devices and device combinations that form the station.
hasPosition	This property defines the position of the entity.
hasOrientation	This property defines the orientation of the entity.

The spatial relations of cells and stations, namely *adjacentTo*, *connectedTo*, and *precededBy* allow spatial reasoning with the digital information. Based on this, it is possible to know in which order the capabilities exist in the system, e.g. in the case of an assembly line where the product flows through multiple stations.

4.5.3. Connection between products, processes, capabilities and systems

The formalization of the product information and related processes was done by Lanz (2010) and will therefore not be discussed in detail in this thesis. However, a simplified illustration of the relations between product, process, capability and resource information is shown in Figure 44.

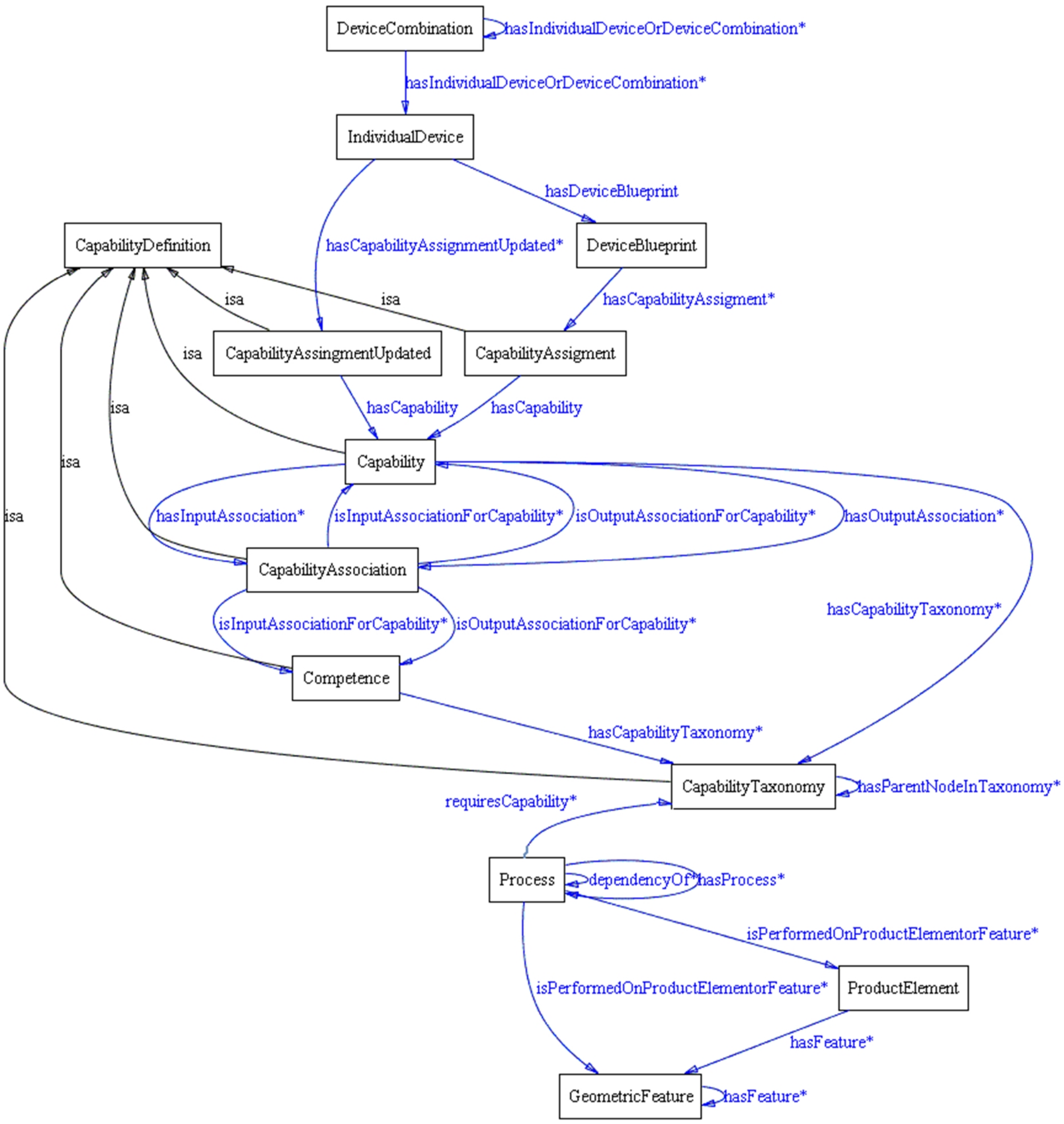


Figure 44. Relations between products, processes, capabilities and devices.

In the proposed capability matching approach, the product requirements and system capabilities are the core concepts, while processes serve as a sort of a middleware between those two. Related to Service Oriented Architecture (SOA), the processes can be considered as services. Products require services (processes) and systems provide services (processes). Processes have basically only two parameters: the time it takes to complete (duration) and their place in the process graph. Everything else can be modelled through the capabilities. This is a completely new way to model the problem. The pre-process plan is modelled in the process domain of the Core Ontology. The properties of the Process class are shown in Table 26.

Table 26. Properties of the Process class.

Property name	Description
requiresCapability	A process requires one or more capabilities. Capabilities can be defined at different levels of detail (e.g. “material removing“ vs. “milling”). This property is used to make the reference to the correct level on the capability taxonomy.
hasSetup	A process may need some set-up activities before it can take place.
isPerformedOn ProductElementOrFeature	Processes are performed on ProductElement or any subclass of ProductElement or they can be performed on GeometricFeature.
requiresResource	A process requires one or many resources. This property can be either pre-defined, or the proper resources can be reasoned out by the capability matching. In the latter case (and in the approach taken in this thesis) this property will be left empty.
dependencyOf	Dependency defines the order in which processes occur. A process can have multiple dependencies, e.g. processes 1, 2, 3 and 4 must be completed before process 5 can occur.
hasDuration	This property defines the duration of the process.
hasProcess	A process can have multiple sub-processes.

4.5.4. Spatial and temporal representation and reasoning allowed by the ontology definitions

There are certain restrictions that limit the cooperative capability of the resource entities. These are their spatial relationship and technological compatibility. If the resources are far away from each other they are not able to cooperate, or if they don’t have compatible interfaces they cannot be combined. Therefore, an important restriction which limits the capability of a combination of devices is their spatial relationship. Resources which are not spatially related cannot cooperate, e.g. a robot on the factory floor and a gripper in store.

When matching the product requirements with the capabilities of the existing system, it is important to know the temporal order of the required capabilities, as well as the temporal order of the provided capabilities. Therefore, some way to describe the layout of the system is needed, in order to know in which order the capabilities appear in the system. This is not crucial in cell-type manufacturing environments, where the product doesn’t need to have a predefined track. On the other hand, in the case of line-type production systems, where the product is moving through the line from one station to the next, this is important.

The spatial and temporal reasoning of the existing resources, capabilities, and needed process activities is enabled by the common relations used in the Core Ontology and summarized in Table 27.

Table 27. Spatial and temporal relations used in the Core Ontology.

Relations	Description
Spatial relations	
is_located_in	Indicates the factory or other higher level resource element where this specific resource element is located.
adjacent_to	Indicates the system components that are adjacent to this specific resource element, but not physically connected.
connected_to	Indicates the system components that are physically connected to this specific resource element.
preceded_by	Indicates the system component that directly precedes this specific resource element. This is used in the system description to indicate the order of the stations or cells in the line layout.
Temporal relations	
dependency_of	Used in a process description to indicate the dependency between the activities, i.e. which activities have to be performed before a specific activity can take place.

The product requirements, defined according to the product features and rule-based reasoning, contain the required capabilities, their parameters and their temporal order. Every activity in the pre-process plan, except the first one, has a predecessor, which is defined by the temporal *dependency_of* relation. For each activity, a reference to the capability taxonomy is defined, and in this way the chain of required capabilities can be expressed. If two capabilities have the same predecessor, they should or can be executed in parallel.

In addition to determining the resources' ability to cooperate, the spatial relations between the resources on a production line layout also determines the temporal relations between the existing capabilities. These can then be compared with the temporal order of the capabilities required by the product.

In addition to those common relations shown in Table 27, the spatial and temporal reasoning is supported by the following properties. The DeviceCombination class holds the information about the combinations of devices. These combinations contain the information about physically connected individual resources. The device combinations can also contain other device combinations. Therefore, the connectivity of the individual resources can be reasoned based on the DeviceCombination class. The DeviceCombination doesn't, however, specify the relative location of the resources in the combinations. Station class holds the information about which individual devices or device combinations belong to a station. In the case of line-type production, the spatial relation of the stations (*connected_to, adjacent_to, preceded_by*) indicates the sequence in which the capabilities occur on the line. In the case of cell type production, the spatial relation of the cells and stations does not restrict the order of processes, assuming that the logistics between the cells and stations is performed manually or by other flexible transportation methods.

In principle, the spatial representation of distance can be either expressed on some "absolute" scale or provide some kind of relative measurement. Because the detailed geo-spatial reasoning is beyond the scope of this thesis, the presented relations between the resource and process entities provides sufficient information for the capability reasoning. Should more detailed representation be needed in the future, the Core Ontology allows the detailed position and orientation of the resources on the factory floor to be saved.

4.6. Adaptation schema

A fundamental enabler for efficient adaptation planning and reactive adaptation is an understanding of the activities and information flows that characterize the process. Therefore, activity models representing the adaptation schema have been developed. The adaptation schema aims to capture the overall adaptation process and activities as well as the information flows, controls and resources required during the adaptation planning and reactive adaptation. It defines how the other components of the developed methodology, namely the resource model, taxonomy and rules for capability matching, are used during the adaptation process.

The definition of the schema has been an iterative process. First, before starting the development of the resource description and capability model and the rule-base for capability matching, an initial sketch of the schema was formulated. After the resource description model and the rule framework had been developed, a final version of the adaptation schema was drawn up in the form of IDEF0 (Integration DEFinition for Function Modelling) activity diagrams. Even though the adaptation schema definition was defined as Sub-objective 1 in the problem definition chapter, it is here presented as third, after the other components of the adaptation methodology. This is mainly due to the iterative nature of its development. The schema also neatly brings together and summarizes the different concepts developed and discussed in the previous chapters, and puts them into a context forming the backbone of the developed adaptation methodology. The following chapters will first discuss the overall view of the adaptation process, followed by the detailed definition of the adaptation schema through the IDEF0 activity diagrams.

4.6.1. Overall view of the adaptation process

Below, some general aspects which are important for the adaptation of production systems will be discussed. These have affected the development of each component of the adaptation methodology, especially the adaptation schema. As is known, changes always incur costs and take time. Therefore, when the requirements targeted at a system change, it is important to try to minimize the changes needed by the system. Consequently, the possibility for logical or parametric adaptation should be evaluated first. The physical adaptation is done only if the current system is not able to accomplish the needed tasks by changing the process parameters of the devices, changing the routing, or changing the programs of the devices. If the needed capabilities exist in the current system, no physical adaptation is needed, but the adaptation is based on allocation and scheduling of suitable resources based on the order information as well as context-specific and user-given criteria. If physical adaptation is required, the adaptation distance, i.e. the magnitude of needed changes, should be minimized by maximizing the number of cells and stations that remain; maximizing the number of cells and stations that remain in their original location; and maximizing the number of devices that remain.

Bi et al. (2008) stated that when reconfiguring an existing production system it is necessary to understand the requirements according to which the original system was designed, and to find out where there are changes to those requirements. In other words, the original design decisions that led to the specification of the existing system should be understood. This indicates that the adaptation could rely on a comparison between the old product and the new one, or the process plan of the old product and the process plan of the new one and through identifying the differences between them, in order to reason out the changes required for the system. The adaptation methodology and schema developed during the course of this thesis work don’t adopt that approach, but rely on comparing the existing system specification with the new requirements, mainly for the following three reasons.

Firstly, the systems, processes and products often go through some changes during their lifecycle. Change in one domain often also causes changes to the other domains. For example, the production system changes from its original specification during its operation time, because of, for example, maintenance, service, malfunctions, etc. These may affect the processes, which may no longer fully correspond with the original process description. However, these changes are not always reflected in the models used to describe those domains. Due to this iterative process, the links between the original design decisions and the system are no longer useable, so the models should be frequently updated in order to keep the links between product specification and the current system valid. In a frequently changing environment, this updating would be a burdensome task. Therefore the approach presented in this thesis proposes to keep the system description updated and compare it with the new product requirements.

Secondly, the current design and information systems are not able to preserve the designer’s intentions. This means that the reasons behind certain design decisions are lost when the model leaves the designer’s table, as was recognized by Järvenpää et al. (2010). Therefore all the links between the original product and the original system specifications may not be understood by anyone other than its original designer.

Thirdly, this thesis also aims to support dynamic adaptation which occurs while the system is running, such as the dynamic routing of the orders based on the available capabilities and product requirements. In this kind of situation it is more important to know the requirements of the current product, rather than its differences to the previous one. In the case of multiple resources with different capabilities that can produce the same feature, the aim is not to use the same resource that was used with the previous product, but to use whichever one is available and has a suitable capability. Therefore, comparing the system specification to the requirements, rather than comparing the new and previous product (or their process plans) is seen as a more feasible approach.

The simplified adaptation schema is illustrated in Figure 45. The simplified schema is modelled in the form of a simple flowchart and is fully incorporated into the activity diagrams that form the full detailed schema, discussed in the next chapter. The adaptation is based on a comparison of the new order requirements with the capabilities of the current production system. The comparison is first performed at the capability concept name level, and the match is subsequently evaluated at the parameter level. Any missing technology is first searched for from the company’s own device libraries and then from the system providers’ libraries. Different configurations are generated with matching devices and compared against user-defined criteria and other adaptation constraints. Finally, the best scenario for the given situation is selected. The change in the order requirements can be caused either by the change in the product requirement, e.g. a new product model or variant, or other changes in the order properties, such as a faster production time.

Particularly in the final phases of the adaptation planning, human-machine interaction is normative. Computer power is used to manage and filter a vast amount of information to create different solution scenarios, while human intelligence is needed to validate these and select the desired one.

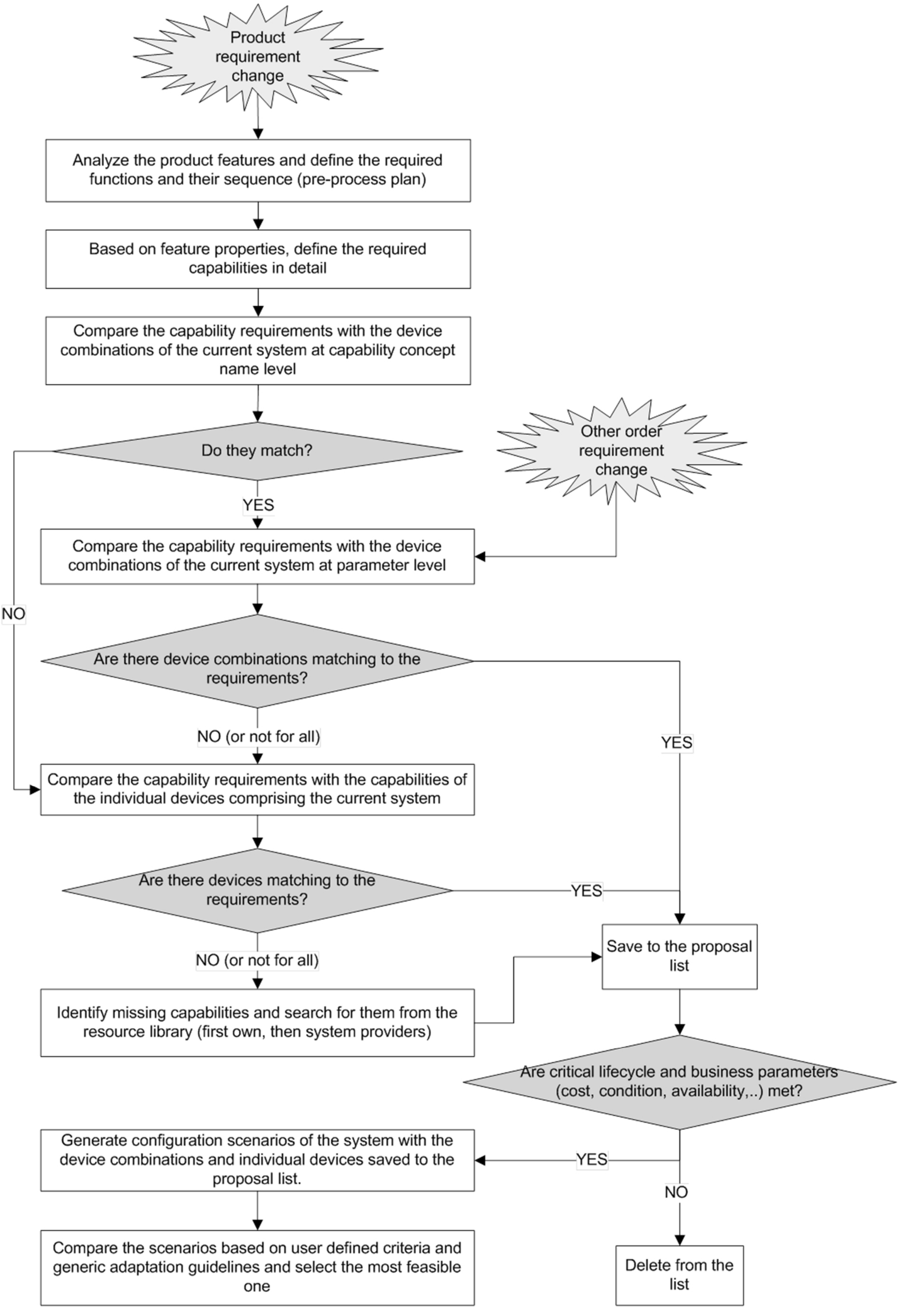

Product requirement change
Analyze the product features and define the required functions and their sequence (pre-process plan)
Based on feature properties, define the required capabilities in detail
Compare the capability requirements with the device combinations of the current system at capability concept name level
Do they match?
YES
NO
Other order requirement change
Compare the capability requirements with the device combinations of the current system at parameter level
Are there device combinations matching to the requirements?
NO (or not for all)
YES
Compare the capability requirements with the capabilities of the individual devices comprising the current system
Are there devices matching to the requirements?
YES
NO (or not for all)
Save to the proposal list
Identify missing capabilities and search for them from the resource library (first own, then system providers)
Are critical lifecycle and business parameters (cost, condition, availability,..) met?
YES
NO
Generate configuration scenarios of the system with the device combinations and individual devices saved to the proposal list.
Compare the scenarios based on user defined criteria and generic adaptation guidelines and select the most feasible one
Delete from the list

Figure 45. Simplified adaptation schema.

As seen in Figure 45, the approach taken here is reactive, rather than proactive. The system is adapted when the external requirements change, and the evolution of the market is assumed to be unknown and cannot therefore be used as additional information to guide the adaptation process. This means that the system is only adapted according to the requirements of the next product. If more information (even uncertain) of the future becomes available, this should be taken into account during the adaptation planning, in order to enable some level of flexibility for the future scenarios to be incorporated into the new system configuration. Terkaj et al. (2009b) have touched this topic with their stochastic programming approach, which aims to support the design of manufacturing system architectures whose level of flexibility is focused on specific production requirements.

4.6.2. Activity diagrams for production system adaptation schema

An activity model captures activities and the corresponding information flows between the activities. In the case of production system adaptation planning, the activity model is a tool to formally model and analyse the activities that are needed during adaptation planning and reactive adaptation, and the information flows and resources that are necessary to perform those activities. The activity diagrams are modelled using IDEF0-language, which is a textual and graphical modelling language for system analysis and specification. It is designed to model the decisions, actions and activities of an organization or system. IDEF0 was derived from a well-established graphical language, a Structured Analysis and Design Technique (SADT), which was published by NIST in 1993. (Knowledge Based Systems Inc. 2010.)

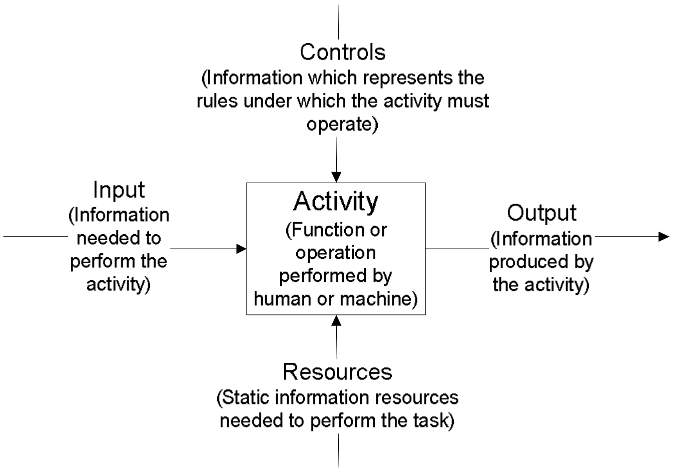

Controls
(Information which represents the rules under which the activity must operate)
Input
(Information needed to perform the activity)
Activity
(Function or operation performed by human or machine)
Output
(Information produced by the activity)
Resources
(Static information resources needed to perform the task)

Figure 46. Elements in IDEF0-diagrams (Knowledge Based Systems Inc. 2010).

Figure 46 presents the elements of IDEF0-diagrams and how they are used in this thesis. Resources, in this context, represent static information resources that the activity may use selectively depending on the needs of the specific situation. These may be tools, information models, software, methods and so on. The word ‘static’ indicates here that the resource is not modified by the activity it takes part in and it remains unchanged over a certain time interval, depending on the resource. Each activity box may be decomposed into its sub-activities in another more detailed diagram. The description of the activities of a system can easily be refined into yet greater detail until the model is as descriptive as is necessary for the decision-making task at hand. (Knowledge Based Systems Inc. 2010.)

In the scope of this thesis all the activities are presented down to the 2nd level of detail. However, most of them could be further detailed. The activities are limited on the digital and virtual knowledge levels, meaning that the physical implementation is beyond the scope of these diagrams. In order to ensure the readability of the activity diagrams, they are placed in the appendices (6 - 9)., The explanations of the used terms indicating the inputs, outputs, controls and resources in the diagrams, can also be found in Appendix 10. This section confines itself to a brief discussion of the activities.

Top level activities

Node A0: Overall adaptation schema

A1 – Definition of product requirements
During this activity, the product requirements affecting the production system design and the requirements for adaptation will be defined.

A2 – Matching product requirements with existing capabilities
This activity concentrates on matching the requirements set by the product with the capabilities provided by the existing system and its components.

A3 – Creating an adaptation plan for the current system
This activity aims to specify how the current system needs to be changed in order to meet the new requirements. The changes can be either physical, logical or parametric.

2nd level activities

Node A1: Definition of product requirements

A11 – Feature recognition
During this activity the features are recognized and classified from the product model.

A12 – Analysis of the features
This activity aims to identify the features that affect the process selection and the sequence of the process activities.

A13 – Definition of pre-process plan
During this activity a preliminary process plan for the manufacture of the product is created based on the identified manufacturing features. The pre-process plan defines the required activities at a high level, e.g. “joining” or “material removing”.

A14 – Definition of required manufacturing capabilities
During this activity the required manufacturing capabilities, including the capability parameters, are specified in more detail based on the required activities in the pre-process plan and the feature properties.

Node A2: Matching product requirements with existing capabilities

A21 – Comparing the requirements with the capabilities of the current device combinations at the concept name level

This activity aims to compare the required capabilities with the capabilities possessed by the combinations of devices in the current system at the capability concept name level. The idea is to see if there already exist device combinations, for example workstations, that are able to fulfil the given requirement without any changes.

A22 – Comparing the requirements with the current device combinations at the parameter level
During this activity, the detailed match between the required and provided capabilities is evaluated. This means that the capabilities matching the requirements at the capability concept name level are compared with the requirements at the capability parameter level. This matching is done, as specified by the capability matching framework, according to the rules in the rule-base and the feature properties.

A23 – Identifying missing capabilities in current device combinations
The purpose of this activity is to define those capability requirements that cannot be fulfilled by the device combinations in the current system.

A24 – Comparing requirements with the capabilities of individual devices in the current system
This activity aims to compare the required capabilities with the capabilities of individual devices existing in the current system. The idea is to see if there are suitable devices in the current system that have capabilities that match with the required simple capabilities and could be used to form new device combinations.

Node A3: Creating an adaptation plan for the current system

A31 – Evaluating the needed adaptation type
This activity aims to specify what kind of adaptation is required in order to obtain the required capabilities. The adaptation can be either physical, logical or parametric.

A32 – Searching for devices from device libraries
The aim of this activity is to search for devices that possess the capabilities missing from the current system. The search is first targeted at the company's own device libraries and, if there is no match, the search can be targeted at the system providers' or rental warehouses' libraries.

A33 – Generating system configuration scenarios
During this activity different configuration scenarios of the suitable devices and device combinations fulfilling the capability requirements are generated. These configuration scenarios represent different options for adapting the system at the physical, logical or parametric levels.

A34 – Selecting the most suitable configuration
This activity aims to identify the most suitable system configuration for the given situation based on the capabilities and availability of the devices, as well as different user-defined and context-specific criteria. After the selection has been made, the actual adaptation actions can take place.

As discussed earlier, the adaptation schema supports both human-centric adaptation planning and dynamic adaptation. In dynamic adaptation, the adaptation is based on the reaction of the system. For example, in the holonic world the autonomous entities make the planning "online", while the system is running. Therefore the application of the schema may vary depending on the situation. This means that, for example, not all the inputs and resources shown in the diagrams may be used in every case.

4.7. Evaluating the impact of change

Change in the order requirements usually leads to changes in the system that tries to meet these requirements. One thing is evident: changes incur costs and take time. Therefore, it is important to be able to estimate the impact of the change, in order to facilitate the estimation of the effort and cost needed to accommodate the changes. It would also help in choosing and prioritizing between different product scenarios requiring changes. Eckert et al. (2004) studied the impact of engineering design changes on the overall product design. They stated that in complex products with closely linked parts, changes to one part of the system often cause changes to another part, which in turn can propagate further change. Similarly, these product changes can impact on the production system components, which can in turn affect other system components. A change that may initially seem small and simple can cause complex modifications to other parts of the system.

The Venn diagrams in Figure 47 shows an example of the requirements for manufacturing different products (A1 & A2 are variants of the same product, while B is a completely different product). When shifting from the manufacture of one product to another, the size of the common requirements zone, or actually the area left out from the common requirements zone, gives an indication of the size of the needed change to the system. For example, it can be assumed that the larger the requirement area of product B which does not intersect with the requirement area of product A1, the larger is the size of the needed change to the system. This relation is not, however, directly proportional. It is close if the system has been originally designed and optimized for the product A1, has no supplementary capabilities, and has not undergone any changes from its original specification, as discussed in Chapter 4.6.1. In this kind of situation the requirement area of the product A1 can be considered to represent the capabilities of the current system. However, if the system has more capabilities than are needed to manufacture the product A1 (e.g. a flexible system), it may be able to manufacture the next product without any physical changes, even if the product requirements have changed.

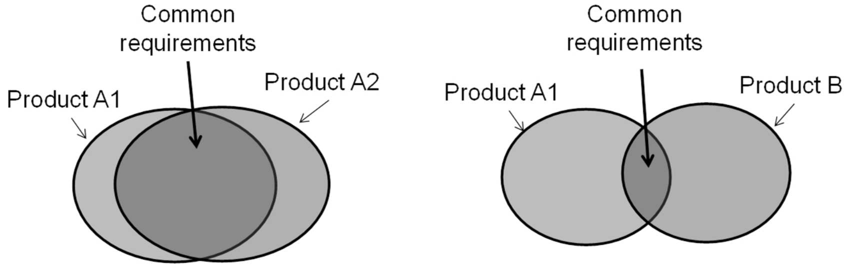


Figure 47. Change in the product requirements causes change to the requirements targeted to the system.

The work conducted during this thesis project, and presented in the previous chapters, makes it possible to recognize the needed adaptation to the system when the product requirements change. Therefore, with this approach, it is possible to evaluate the impact of change from the hardware point of view. Although the developed methodology concentrates only on defining the required changes to the system hardware, it also opens up the possibility to roughly estimate the time and money that need to be invested by providing the required input information in terms of new capabilities that need to be added to the system. Therefore, it facilitates further calculations and decision-making regarding adaptation.

The following chapters present the preliminary concept and formulation of a compatibility domain approach, for estimating the impact of changes in the product requirements on the existing production system. The compatibility domain approach examines the topic from three aspects: the compatibility of the existing system to the new requirements; the relative effort entailed in making the needed modifications to the system; the utilization (re-usability) of the existing system for the new requirements. As the objective was to think up a preliminary approach to evaluating the impact of change, only very simple and coarse formulations of the compatibility, effort and utilization estimation were made, to illustrate, on a conceptual level, how the accomplishments achieved during this thesis contribute to such an analysis.

4.7.1. Compatibility domain approach - Evaluating the compatibility, effort and utilization

As stated earlier, the compatibility domain approach aims to investigate the impact of changes from three different perspectives, namely compatibility, effort and utilization. The meanings of these terms are specified in Table 28. A graphical representation will be given for each of these perspectives. These will be discussed in the following paragraphs.

Table 28. Terms used in the compatibility domain graphs.

Term	Definition
Compatibility	Defines what proportion of the product's capability requirements (in %) can be satisfied by the current system.
Requirement change	Presents the change in the product requirements targeted at the system (i.e. the capability requirements).
Utilization	Defines what proportion of the current system's capabilities (in %) can be utilized for producing the new product.
Effort	Defines the relative amount of new capabilities (compared to the existing system capabilities) that must be added to the current system in order to be able to produce the new product. It only considers the effort of making the system compatible with the new requirements, not removing the unused capabilities.

Compatibility

Compatibility aims to present how compatible the current system is with the new product requirements in terms of its capabilities. It represents the relative amount of the given product requirements that can be satisfied with the existing system in its current configuration. The configuration consists of physical, logical and parametric aspects. The term "compatibility" doesn't refer to an optimal solution for the given requirement, but to an intersection, where the requirements and capabilities match and the system can continue its operation. If the system can satisfy all the capability requirements, the compatibility is 100%. If the needed capabilities exist in the current system, but just a few parameters or programs need to be adjusted, the compatibility is near 100 %, assuming the required parameters are in the range of the existing capability parameters. If physical changes to the system are required, the compatibility will be lower. The capability parameters may have a certain range, or they may be fixed. For example, in the case of "drillBit" the drilled hole will have only one possible diameter, whereas the drill itself is usually able to hold and spin different sizes of drill bits and screwing heads. In other words the “spinningTool” capability parameters have a certain range that the capability can handle.

In order to roughly estimate the compatibility of the existing system to the given new requirements, and to get quantitative values for comparison, the following factors defined in Table 29 are considered.

Table 29. Factors used for evaluating the compatibility of the production system to the requirements.

Factor	Definition
ReqCaps(x)	Number of capabilities required by the product *x* (1 capability = 1 point)
ExistCaps(x,y)	Number of existing capabilities in system *y* that fulfil some of the capability requirements of product *x* and are useable in the given situation (1 capability = 1 point)
AllSystemCaps(y)	Number of all the capabilities that exist in system *y* (1 capability = 1 point)
PhysicModifCaps (x,y)	Number of existing capabilities in system *y* that can be modified by small physical changes to be compatible with the requirements of product *x*, i.e. changing the simple capabilities it is composed of. These include only small changes, like changing the tool in the lathe. If the main device in the combination (lathe) needs to be changed, this won’t apply. (1 capability = 0.5 points)
ParamModifCaps (x,y)	Number of existing capabilities in system *y* that can be modified by parametric or logical changes to be compatible with the requirements of product *x*, e.g. changing the speed of the robot. (1 capability = 0.75 points)
NewReqCaps(x,y)	Number of capabilities required by product *x* that don’t exist in the system *y* (1 capability = 1 point)

The “PhysicModifCaps” and “ParamModifCaps” are included in the formula in order to enable small adaptation actions, such as changing a tool or program to be taken into account. The coefficients 0.5 and 0.75 are adopted for capabilities requiring small physical changes or parametric changes, respectively, in order to take into account, at a high-level, the fact that such capability modifications are less cumbersome than large physical adaptations, like changing complete machines. In other words, they aim to give a very much simplified, rough estimate of the ease of the adaptation action in question. Other than that, the formula doesn’t distinguish the different capabilities that need to be adapted, but they all have the same value, regardless of their importance, size, price or any other parameter. Therefore all the changes have the same value in the graph, even though, in reality, some changes are more difficult, time consuming and costly than others. Because these are impossible to estimate without real measured values and experiences, as noted in the industrial reconfiguration and re-use scenarios, the proposed compatibility domain approach only takes into account the ratio between the amount of required and provided capabilities. If more educated evaluations about the relative size of the required capability adaptation are available, other coefficients than 1, 0.75 and 0.5 can be used in the formulas.

The compatibility of system *y* for the product *x* is calculated by the following formula:

$$Compatibility(x,y) = \frac{ExistCaps(x,y) + 0.5 * PhysicModifCaps(x,y) + 0.75 * ParamModifCaps(x,y)}{ReqCaps(x)} * 100\ \% \tag{2}$$

In the “compatibility – requirement change” graph, the x-axis represents the change in the product requirements targeted at the production system. The propagation of the changes is taken into account as discussed in the “magnitude of the required change to the system” -section. In the Venn diagrams, seen in Figure 47, the requirement change is represented by those areas which are outside the intersection of the two compared product requirements. The requirement change between two subsequent products, product A and B, is calculated as shown by the formulas (3), (4) and (5). It is assumed that the current system y is fully compatible with product A and no supplementary capabilities exist (ReqCaps(A) = AllSystemCaps(y)).

Capability requirements of product A, which are different from the capability requirements of product B:

$$\begin{aligned}ReqCaps(A)not(B) &= ReqCaps(A) - (ExistCaps(B,y) + 0.5 * PhysicModifCaps(B,y) +\\ &\quad 0.75 * ParamModifCaps(B,y)) \qquad (3)\end{aligned}$$

New capability requirements of product B, which are different from the requirements of product A:

$$\begin{aligned}ReqCaps(B)not(A) &= NewReqCaps(B,y)\\ &= ReqCaps(B) - (ExistCaps(B,y) + 0.5 * PhysicModifCaps(B,y) +\\ &\quad 0.75 * ParamModifCaps(B,y)) \qquad (4)\end{aligned}$$

Requirement change between product A and B:

$$\begin{aligned}Requirement\ change(A,B) &= ReqCaps(A)not(B) + ReqCaps(B)not(A)\\ &= ReqCaps(A) -2 * \left(ExistCaps(B,y) + 0.5 * PhysicModifCap(B,y) +\right.\\ &\quad \left.0.75 * \ ParamModifCaps(B,y)\right) + ReqCaps(B) \qquad (5)\end{aligned}$$

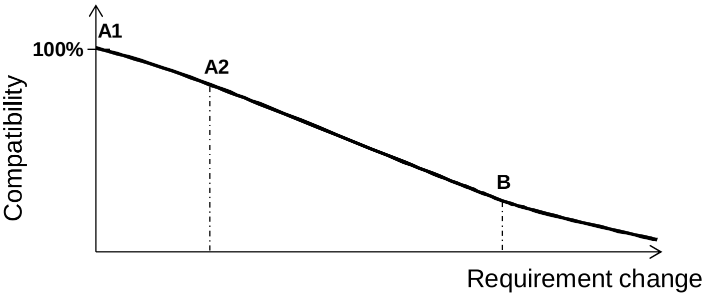


Figure 48. Conceptual view of the compatibility versus requirement change.

Figure 48 shows an example of a “compatibility – requirement change” graph. The graph represents a conceptual view of the compatibility of the system to the requirements as changes in the requirements grow. The graph assumes that the system is fully compatible with the requirements set by product A1, and that it has no extra capabilities, i.e. the system was originally developed for product A1. In principle, the compatibility graph is a descending curve when the subsequent products are relatively similar, i.e. having similar amounts of required capabilities. However, the shape of the curve should not be generalised as it is easy to come up with special cases which won’t fit this descending curve.

The greater the compatibility, the more requirements can be satisfied with the current system and the fewer the changes which are needed to the existing system. However, again, this comparison is valid only with very similar products, i.e. products having similar amounts of required capabilities. For example, two products may have the same compatibility, but completely different effort to modify the system to be able to produce the whole part. This is the case, for example, when product B has 20 required capabilities, of which 10 are satisfied, versus product C, with 2 required capabilities of which 1 is satisfied. In both cases the compatibility is 50%, but the first one needs much more adaptation effort to be actually able to produce the product (10 new capabilities to be acquired for product B versus 1 capability for product C). The "compatibility – requirement change" graph is therefore only able to analyze what proportion of the product requirements can be satisfied in different cases, but not the actual effort of adapting the system to be fully compatible with the product requirements.

If the new product has fewer requirements, none of which are additional to those needed for the original product, and the remaining requirements have similar capability parameters, the system is definitely able to cope with those new requirements. Therefore the requirement change can be considered to be negative and the compatibility of the system is 100 %. However, it needs to be noted that even if the compatibility of the system is 100 %, it may not be the most optimal way to produce the product. For example, if the production volume of this new, simpler product is high, it may not be cost efficient to occupy the whole system with excess capability in order to manufacture that product.

Magnitude of the required change to the system (effort)

The previous graph, Figure 48, doesn't indicate the effort needed to adapt the system to the new requirements. Another graph needs to be drawn for this. The relative effort of an adaptation is here estimated as the relation between the missing capabilities (capabilities to be acquired) and all the existing capabilities in the system. In other words the effort is defined purely based on the new capabilities that need to be added to the system, and no other factors, such as ramp-up time and cost, are taken into account. The formulation of effort enables the comparison of different product scenarios based on the required adaptation effort. An example of this can be seen in Figure 49, where the product scenarios are drawn on the graph according to their compatibility and effort. This example is based on the data presented in Chapter 5.3.4. The graph clearly shows that compatibility doesn't necessarily indicate the needed effort as discussed above (see products E and G).

Effort of making the current system y compatible with the new capability requirements of product B:

$$Effort(B, y) = \frac{NewReqCaps(B,y)}{AllSystemCaps(y)} * 100\% \qquad (6)$$

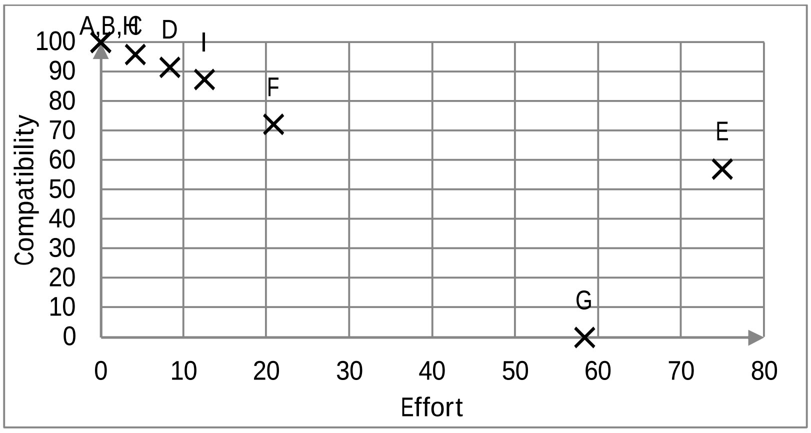


Figure 49. Compatibility versus effort – example.

The procedure and complexity of an adaptation are not only strongly dependent on the type of change in the requirements, but also on the architecture of the system (e.g. line-type production or production cells). The magnitude of the required change to the existing system is also highly dependent on the differences between the characteristics of the previous and the new product. For example, if new kinds of processes are required, some new technology needs to be added to the system (assuming the system was originally designed for the previous product). If the required processes don't change, but the parameters are different, the current system may be able to meet these new requirements simply by changing some of the parameters.

The type, architecture and degree of automation of the current system also affect the magnitude of the needed adaptation. For example, dealing with machining systems or automatic assembly systems are two completely different issues. For instance, if the product dimensions change, in machining it may just mean that the amount of material to be removed changes and the tools may remain the same. Or, perhaps just the tool needs to be changed. As long as the billet size doesn't exceed the maximum size that can be machined with the machine, the product can usually be manufactured with only small modifications to the system. In the case of an assembly system, changes in the dimensions and geometry of the product and parts may have a strong impact on the system. The feeders may need to be changed, the pallets and grippers may need to be redesigned, the robot reprogrammed and so on. Therefore, when formulating the graphs and defining the new requirements, the propagation of the changes to the system requirements have to be taken into account. The value of new requirements cannot, therefore, be defined purely by the differences in the product characteristics, but the current system architecture also needs to be taken into account. For example, if a new process is required in addition to the old ones, new technology needs to be integrated into the existing system. Depending on the system architecture, this may result in more or less other requirements being placed on the system, which affects the overall magnitude of the required change. By determining the changes in requirements in terms of the existing system, the propagation of these changes can be taken into account.

The system architecture also has another kind of effect on the process and magnitude of the adaptation. For example, if the production volume changes, the current system needs to be evaluated based on the new target cycle time, assuming that increasing the number of work shifts is not an option. In the case of a line-type layout, underperforming stations need to be identified and possibly replaced in order to balance the line and achieve the desired cycle time. The whole line may need to be reorganized and tasks re-allocated, causing significant changes to the system. In contrast, in cell type production it is easier to add new resources to the underperforming cells, or to use more shifts in the slower cells and make buffers.

The "compatibility – effort" graph indicates the relative magnitude of change needed to the system and therefore gives some indication of the costs of the adaptation. However, it still cannot be regarded as an accurate estimate, because the effort, cost and time needed for the adaptation is affected by many other parameters as well. For example, the number of required changes may be similar in two completely different cases, one requiring the replacement of devices, and the other requiring the addition of new devices. In the latter case, it is also likely that there is a requirement for additional space, which may need to be acquired. Therefore, in the latter case, the effort and cost incurred by the change may be greater. Thus, the "compatibility – effort" graph doesn't alone provide enough information for estimating the effort and costs of changes. It is also important to consider whether all those changes can be implemented with in-house resources, or whether new resources need to be acquired. This drastically affects the cost of the change, and probably also the time needed to accomplish it. Also the ramp-up may differ from system to system, and this is not considered here.

Even though it might seem slightly inconsistent that the effort is determined without considering the unused capabilities, there is a reason for this. The approach taken in this thesis aims to serve situations where the next order (the product to be produced and the required quantity) is unknown. Therefore, it is not necessarily wise to remove the unused capabilities from the system, because they may be needed for the next order. This is especially the case when the order sizes are small. Basically, the cost of occupying the unneeded capabilities depends on the lot size, the product manufacturing time, the situation, the order book and so on. How long will the system be occupied? What products need to be produced next? Do they need the capabilities that are about to be removed from the system? Could the unneeded devices be used in other stations at the same time? The incorporation of such factors into the method should be considered in the future in order to allow the knowledge or assumptions regarding future product scenarios to be taken into account during the adaptation planning.

Utilization of the current system

Utilization of the current system shows how much of the current system can be utilized (re-used) to produce the new product.

Utilization of the current system y for producing product B is calculated as follows:

$$Utilization(B,y) = \frac{ExistCaps(B,y) + 0.5 * PhysicModifCaps(B,y) + 0.75 * ParamModifCaps(B,y)}{AllSystemCaps(y)} * 100\% \quad (7)$$

Based on the previous results, a “compatibility versus system utilization” graph can be drawn, as shown in *Figure 50*. It allows the simultaneous estimation of how much of the product requirements can be satisfied with the current system, and how much of the current system’s capabilities can be re-used without any modifications, in different product scenarios. Considering the system utilization is especially important if large volumes are produced. For example, it would be wasteful to occupy the whole system for a long period of time if only a few of the system’s capabilities are being utilized. On the other hand, the profitability of the currently unneeded flexibility depends on the future product scenarios.

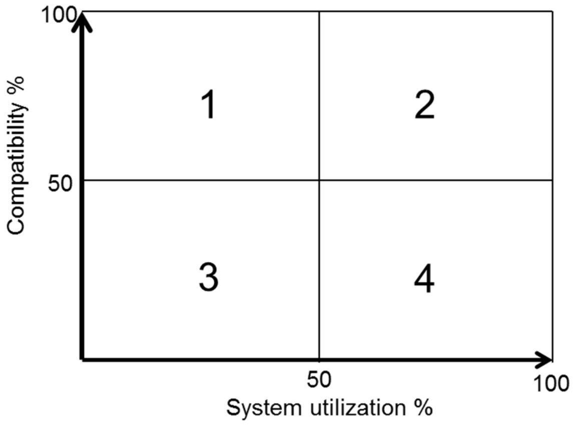


Figure 50. Compatibility versus system utilization.

The product scenarios analysed with this graph can be compared based on the four quadrants drawn on the graph:

1) Those product scenarios falling into quadrant 1 have strong compatibility with the current system, but the system has a lot of capabilities that are not utilized. Therefore these scenarios may be costly.
2) Products in quadrant 2 have strong compatibility with the current system and most of the system capabilities are utilized. This gives a rough indication that these products are the most feasible ones.
3) Product scenarios falling into quadrant 3 exhibit both weak compatibility and weak utilization and therefore they don't show much potential for re-using and adapting the existing system.
4) Product scenarios falling into quadrant 4 have weak compatibility with the existing system, but the system utilization would be strong. This indicates that the existing system could be extended to meet the new requirements by adding new resources.

A practical example of utilizing the compatibility domain approach can be found in Chapter 5.3.4.

4.7.2. Discussion of the compatibility domain approach

There are many different methodologies available for evaluating the impact of changes. In general though, these view the topic from the product development perspective, and evaluate the effects of engineering changes during the product's lifecycle or the propagation of changes within the product structure, as in Eckert et al. (2004). Other viewpoints taken by researchers are, for example, the impact of engineering changes on material planning, cf. Wanström and Jonsson (2006), the impact on costs, cf. Oduguwa et al. (2006) and the impact on the design process, cf. Terwiesch and Loch (1999). Attempts have also been made to evaluate and control the impact of product design changes on the production system using, for example, Design for Assembly (DFA) and Design for Manufacturing (DFM) methodologies (Boothroyd et al. 2002). Tolio et al. (2010) presented an extensive review of approaches which try to tackle the co-evolution of products, processes and systems from different perspectives. The compatibility domain approach presented here concentrates particularly on evaluating the impact of changes in the product and order requirements on the production system in terms of the new capabilities that need to be added to the system. It is not intended to replace the existing approaches, but to complement them by viewing the problem from a different angle. Investment decisions are not covered by this approach, yet the compatibility domain approach does provide valuable input data for such decisions. This is because it is only after the impact of change has been evaluated that the time, cost and resources can be allocated (Eckert et al. 2004).

The compatibility domain approach viewed the impact of changes from three viewpoints, namely compatibility, effort and utilization. The first views the problem more from the product perspective, whereas the latter two view it from the system perspective. The last two diagrams facilitate the comparison of different product scenarios in regard to the needed adaptation effort and the re-usability of the system.

This compatibility domain approach utilizes the resource capability descriptions and the capability matching framework presented earlier in this thesis, in order to identify those resources or points in the system where adaptation activities are needed. Changes in these resources may again cause changes in other resources – something which was not directly caused by the original product requirement change, but is a kind of "waterfall effect", typical of interrelated system components.

This propagation of changes has been studied in the literature, especially by Eckert et al. (2004), from the product design perspective. It would also be very interesting to see an extensive research study of this important topic from the production system perspective. Due to the breadth of the topic, there is no detailed analysis of the issue in this study. Here, the emphasis has rather been on finding the intersection where the product requirements and the current system capabilities match, and evaluating any differences. When there are relatively few changes, it is easy to do this reasoning manually, but when there are thousands of changes, it becomes a cumbersome task. Therefore the capability-matching approach developed here is a valuable tool for rapidly evaluating the magnitude of changes needed to the system.

It is important to note the following restrictions to the presented compatibility domain approach:

Restriction 1:
The values given by the formulas (2), (6) and (7) of compatibility, effort and utilization by no means represent absolute values for these three concepts. Instead they provide an estimate of the relative effects of change, and are intended to facilitate a rough comparison between different product scenarios. They help to evaluate the effect of different product decisions on a system within a product family. In order to evaluate the real effort and cost of the required adaptation, the nature of the specific adaptation activities needs to be considered. Currently, this approach evaluates only the relative amount of the required adaptation activities, differentiated through both the large and small physical changes, as well as the logical and parametric adaptation actions. The required effort is defined based on the new capabilities that need to be added to the system, and no other factors, such as system ramp-up time and cost, are taken into account at present.

Restriction 2:
One major issue at the start of the compatibility formulation is the granularity (i.e. level of detail) of the representation of the required and existing capabilities. It is important that the granularity of the capabilities is similar in the scenarios under comparison, in order to ensure the validity of the comparison.

Restriction 3:
The validity of the comparison of different product scenarios with the "compatibility – requirement change" graph depends on the similarity of the compared products.

Restriction 4:
The application of the compatibility domain approach, presented in the case study chapter, is fully controlled by humans, and therefore highly subjective. Therefore, even though the graphs give quantitative estimates of the compatibility, effort and utilization, the input information used to draw the graphs is highly qualitative. Therefore, the estimates are more qualitative than quantitative.

5. VALIDATION OF THE RESULTS – CASE STUDIES

Two case studies demonstrating the validity of the developed adaptation methodology were carried out. These case studies cover both the dynamic and static adaptations, as discussed in the problem description in Chapter 2.2.

Case 1: Holonic production system – Dynamic adaptation based on reaction
This case study was carried out as part of a wider demonstration of the Tekes-funded KIPPcolla (Knowledge Intensive Product and Production Management from Concept to Re-cycle in a Virtual Collaborative Environment) and CSM-hotel (Competitive Sustainable Manufacturing Hotel) projects. The implementation consists of modular ICT-architecture and hardware implemented in the TUT machining laboratory. This represents an open production environment where the adaptation takes place during the runtime of the system by reaction. It is a manufacturing system where the production is not predetermined, for example, the product models and volumes are constantly changing and the system needs to adapt itself "on the fly". The product can be produced with multiple alternative resources and the orders are routed based on resource availability and other relevant parameters. Thus, the adaptation actions of this system relate mainly to finding suitable capabilities from the current system and routing the orders to the available and capable resources based on the reactive self-organizing abilities of the system. The demonstration was the result of the work of multiple researchers. The resource description, capability model and capability-matching framework – the contributions from this thesis work – form an essential part of the demonstration.

Case 2: TUT microfactory – Static adaptation based on planning
This case study was developed purely for the purpose of this thesis in order to illustrate the use of the developed methodology and capability matching rules for the adaptation planning of more static systems. In this case, the adaptation takes place "offline" while the system is not running and it is based on human-controlled planning. This case represents a "traditional" production line, which is relatively stable in the short term with regard to product models and volumes. It is fairly similar, for example, to automotive assembly lines where often only one product is produced on the line and the process plan is predetermined, which means that the product has a predetermined track on the line.

The following chapters will present these two case studies. First, though, the implementation of the modular ICT-architecture supporting these case studies will be discussed. While the case studies validate the adaptation methodology, especially the capability model and the resource description, with regard to their usability to industrial use cases, the ICT-architecture, and in particular the Capability Editor tool, validate the resource description and capability model with regard to their applicability to a formal representation.

5.1. Modular ICT-Architecture

The modular software system architecture was designed to serve the holonic production environment, and is therefore discussed here mainly from that perspective. However, it also supports more traditional human-centric planning, where the different software modules are used to aid humans in the different phases of the adaptation planning process. The software system architecture, illustrated in Figure 51, has several different interoperating software modules, each providing one or two essential functions for the overall system. The architecture follows the dynamic modularization principles, being designed in such a way that each of the modules can be replaced with a new module, if needed, without disrupting the whole system. The interoperability of the modules is mainly based on the shared information model and common knowledge representation, the Core Ontology, and modular services. Each of these modules requires specific domain-related information and by processing this information they provide a set of services for the overall system. (Järvenpää et al. 2011b; Järvenpää et al. 2012a; Lanz et al. 2011; Lanz et al. 2012.)

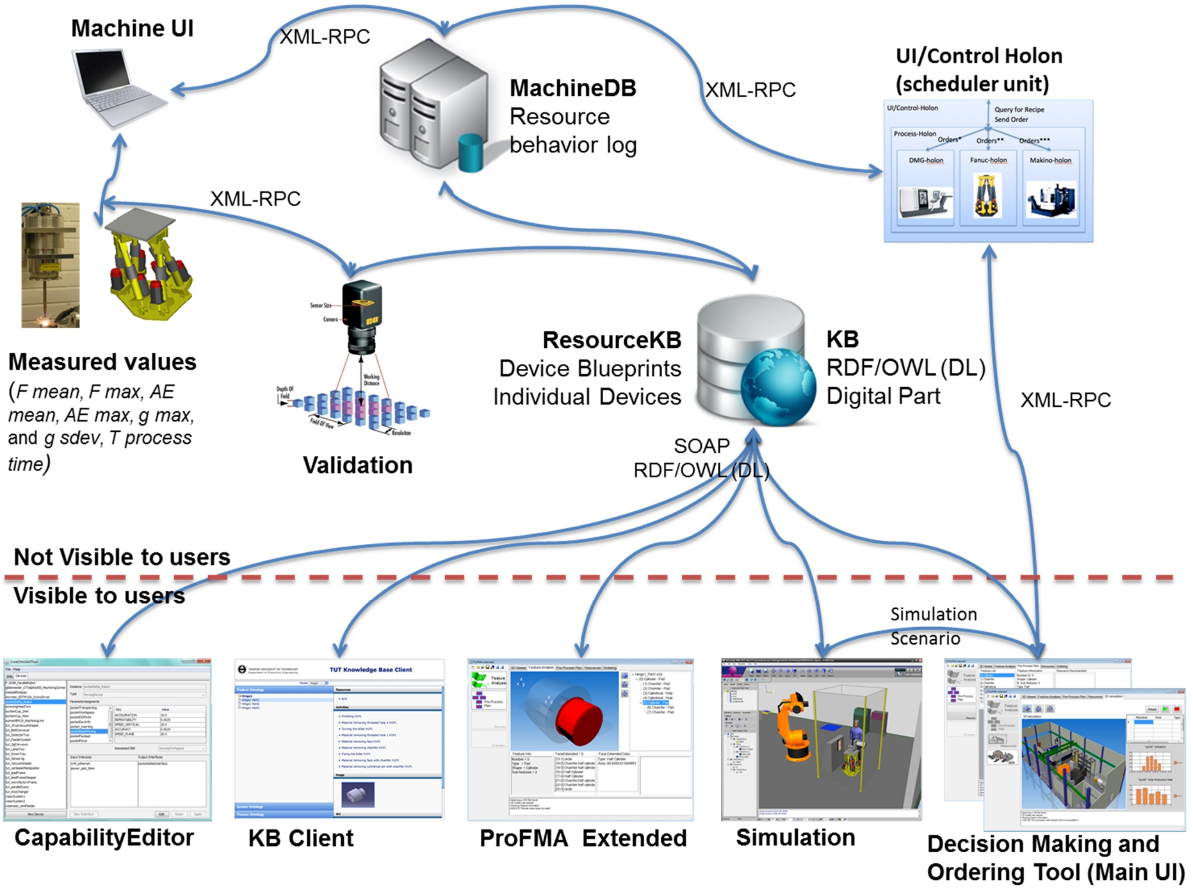


Figure 51. Modular software system architecture, modified from (Lanz et al. 2012).

One essential software module in the ICT-architecture is the Capability Editor, which is designed for assigning capabilities to resources and adding new capabilities and resources to the Knowledge Base. As it plays an important role in this thesis in that it validates the developed resource description and capability model, its use will be discussed in the next section in more detail. Later, the other modules forming the modular ICT-architecture will be briefly introduced.

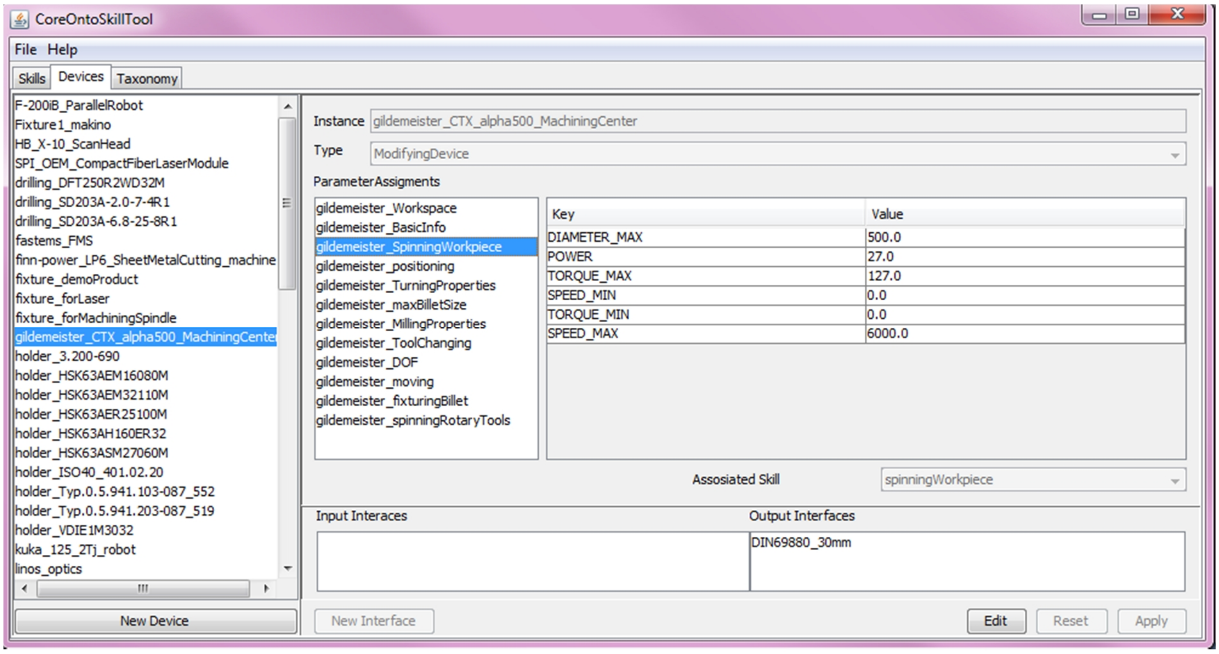
5.1.1. Capability Editor

Unlike the traditional approaches to resource description, the properties of the devices are given as the parameters of the capabilities the devices provide, rather than as properties of the devices themselves. Adopting this new approach requires a change of mind-set, but the resource description task can be simplified using a wizard tool that guides the human through the properties-filling process, or it can be done automatically from standard resource descriptions.

In order to ease the process of adding devices and capabilities to the Knowledge Base (KB), a Java-based Capability Editor was developed. It is not a wizard, but it provides a more user-friendly approach to adding the capabilities and new resource descriptions to the KB, compared with the Protégé ontology editor. The Capability Editor allows a user to add new devices to the ontology and assign them capabilities and capability parameters. It also allows new capabilities to be added to the ontology and enables associations to be created between the simple and combined capabilities. Basically, the editor complies fully with the logic of the capability description discussed in Chapter 4.2. Two use cases for the Capability Editor are described below, namely adding new resources and new capabilities to the KB.

Adding new resources to the Knowledge Base

The Capability Editor allows the user to add new instances of resources to the Knowledge Base. In the Capability Editor, the user can specify the name and type of the resource and describe its mechanical, control and energy -related interfaces. The main task when describing the resources is to assign them capabilities. The user will select the pre-defined, simple generic capabilities from the Capability Editor's capability pool and will then define the resource specific parameters for the capability, making the capabilities unique to each resource. If a suitable generic capability is not pre-defined in the capability pool, a new capability has to be first defined, and then assigned to the resource (see the other use case). Figure 52 shows the user interface of the Capability Editor for adding resources to the KB. In the ParameterAssignments box, the resource specific instances of the generic capabilities are created and values are given to the parameters.



F-200iB_ParallelRobot
Fixture1_makino
HB_X-10_ScanHead
SPI_OEM_CompactFiberLaserModule
drilling_DFT250R2WD32M
drilling_SD203A-2.0-7-4R1
drilling_SD203A-6.8-25-8R1
fastems_FMS
finn-power_LP6_SheetMetalCutting_machine
fixture_demoProduct
fixture_forLaser
fixture_forMachiningSpindle
gildemeister_CTX_alpha500_MachiningCente
holder_3.200-690
holder_HSK63AEM16080M
holder_HSK63AEM32110M
holder_HSK63AER25100M
holder_HSK63AH160ER32
holder_HSK63ASM27060M
holder_ISO40_401.02.20
holder_Typ.0.5.941.103-087_552
holder_Typ.0.5.941.203-087_519
holder_VDIE1M3032
kuka_125_2Tj_robot
linos_optics

Instance gildemeister_CTX_alpha500_MachiningCenter

Type ModifyingDevice

ParameterAssigments

gildemeister_Workspace
gildemeister_BasicInfo
gildemeister_SpinningWorkpiece
gildemeister_positioning
gildemeister_TurningProperties
gildemeister_maxBilletSize
gildemeister_MillingProperties
gildemeister_ToolChanging
gildemeister_DOF
gildemeister_moving
gildemeister_fixturingBillet
gildemeister_spinningRotaryTools

Key	Value
DIAMETER_MAX	500.0
POWER	27.0
TORQUE_MAX	127.0
SPEED_MIN	0.0
TORQUE_MIN	0.0
SPEED_MAX	6000.0

Assosiated Skill spinningWorkpiece

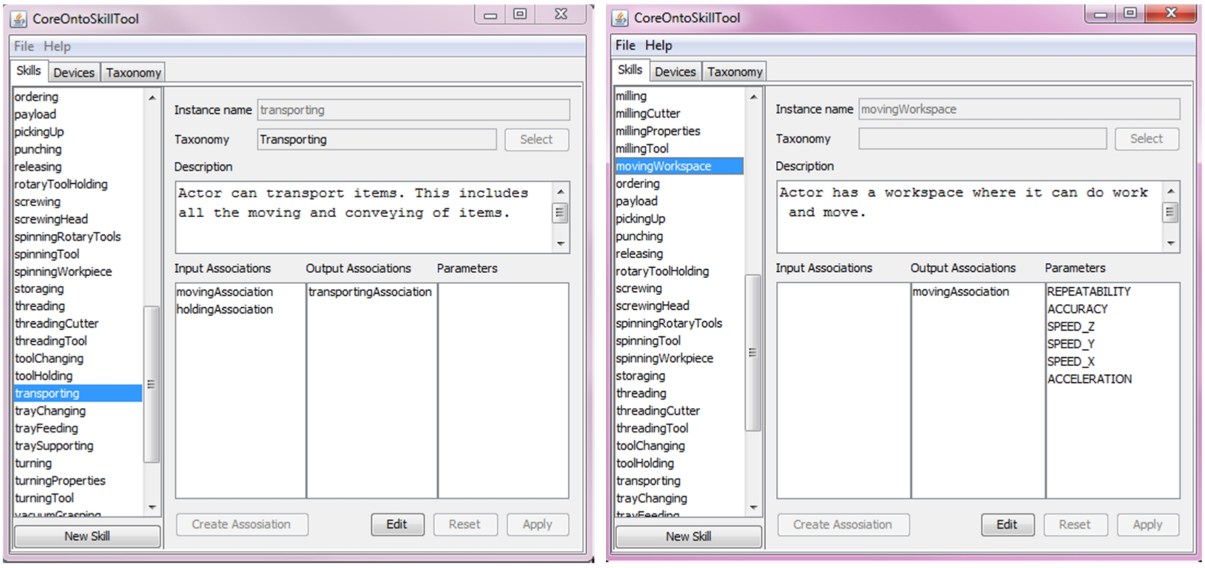
Input Interaces

Output Interfaces DIN69880_30mm

Figure 52. Capability Editor – Adding resources and their capabilities to the Knowledge Base.

Adding new generic capabilities to the Knowledge base

If suitable generic capabilities are not pre-defined in the KB, the user needs to first define them with the Capability Editor. A definition of the generic capabilities includes a definition of the capability concept name and the capability parameters. In addition, the associations between the simple and combined capabilities are created, as discussed in Chapter 4.2.3. Furthermore, in the case of the strong capabilities, a link to the capability taxonomy will be created. Figure 53 shows the user interface for adding capabilities with the Capability Editor.



ordering
payload
pickingUp
punching
releasing
rotaryToolHolding
screwing
screwingHead
spinningRotaryTools
spinningTool
spinningWorkpiece
storaging
threading
threadingCutter
threadingTool
toolChanging
toolHolding
transporting
trayChanging
trayFeeding
traySupporting
turning
turningProperties
turningTool
vacuumGrasping

Instance name: transporting

Taxonomy: Transporting

Description:

```
Actor can transport items. This includes
all the moving and conveying of items.
```

Input Associations	Output Associations	Parameters
movingAssociation holdingAssociation	transportingAssociation	

milling
millingCutter
millingProperties
millingTool
movingWorkspace
ordering
payload
pickingUp
punching
releasing
rotaryToolHolding
screwing
screwingHead
spinningRotaryTools
spinningTool
spinningWorkpiece
storaging
threading
threadingCutter
threadingTool
toolChanging
toolHolding
transporting
trayChanging
trayFeeding

Instance name: movingWorkspace

Taxonomy:

Description:

```
Actor has a workspace where it can do work
 and move.
```

Input Associations	Output Associations	Parameters
	movingAssociation	REPEATABILITY ACCURACY SPEED_Z SPEED_Y SPEED_X ACCELERATION

Figure 53. Capability Editor – Adding new capabilities to the Knowledge Base.

5.1.2. Information flows between the different software modules

Figure 54 is a simplified representation of the information flows between the software modules which form the ICT-architecture. The communication between the Knowledge Base and the modules is done with RDF and XML messages (depending on the situation) using SOAP. The communication between the DeMO tool, UI/control holon and the machine UIs is done using XML-RPC calls. In what follows, the roles of the different software modules in the dynamic operation environment will be discussed one by one. As the Capability Editor was already introduced in the previous chapter, it is excluded here.

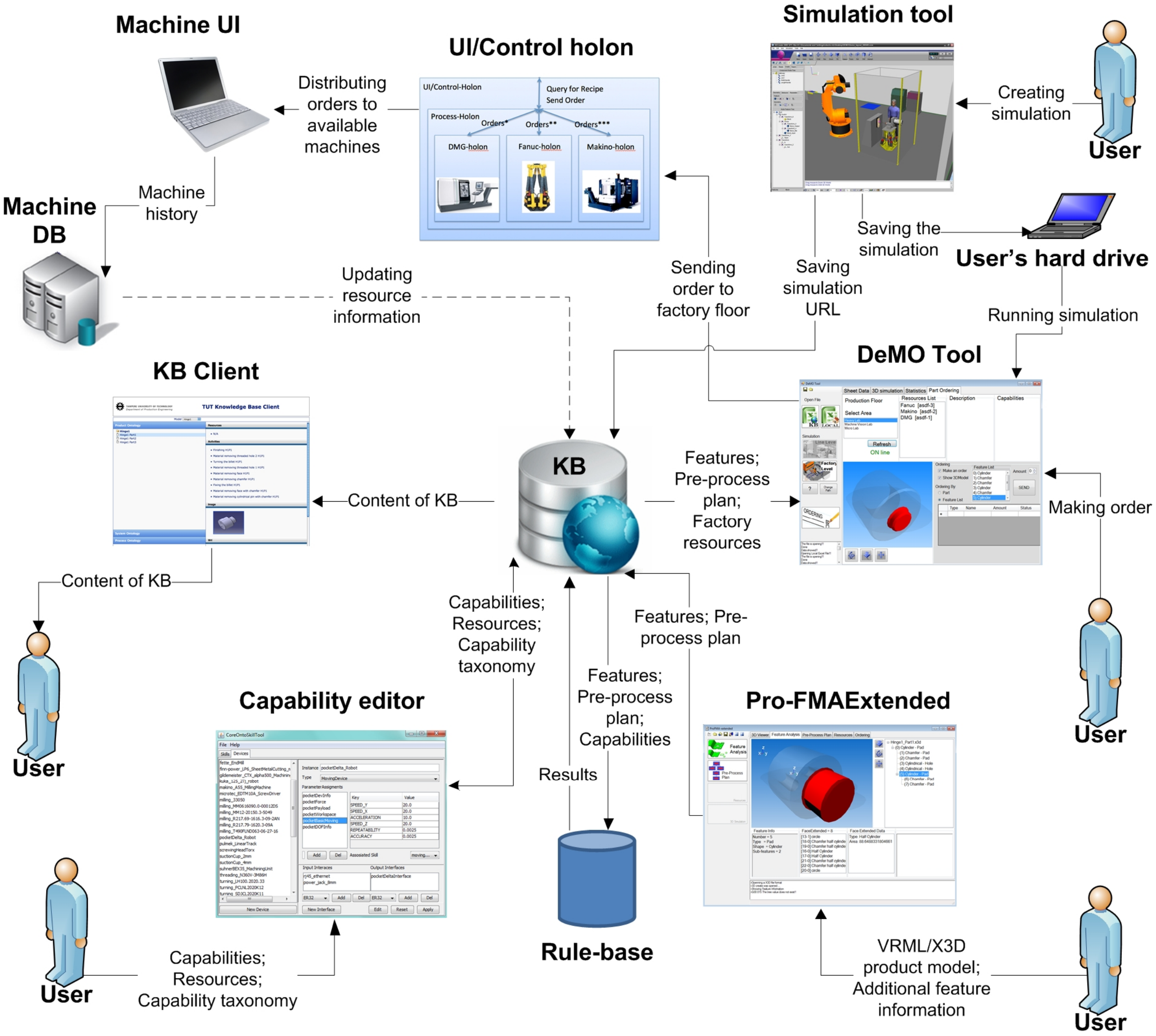


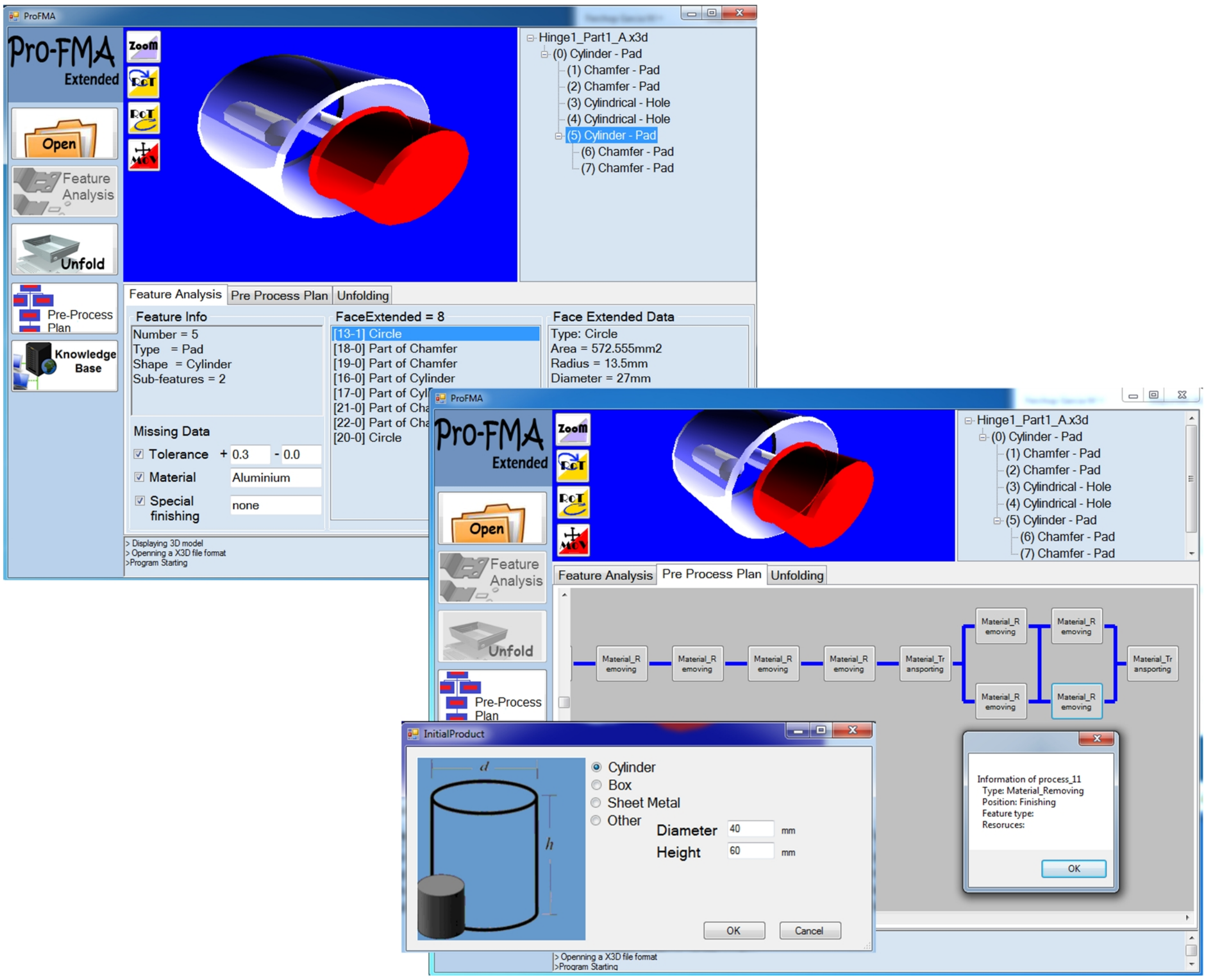
Figure 54. Modular ICT-architecture and simplified representation of the information flows during the operation of the dynamic operation environment.

KB and KB Client

The Knowledge Base (KB) developed in an earlier work by Lanz (2010) and Lanz et al. (2010) was designed to be a system where the data can be stored and retrieved by different applications ranging from the product and process design to the simulation. The following tools and technologies were used to facilitate the approach: an Apache Tomcat web server, an Apache axis 2 web service engine, a Jena semantic web framework, a Pellet reasoner, and a Postgre database. In the KB, the information related to the products, the processes and the systems and their capabilities is saved in the form of a formal ontology, the Core Ontology, modelled with a Protégé ontology editor. As an important part of the overall ICT-architecture, the KB stores the information created by the other software modules, a Pro-FMA Extended, a Capability Editor and a DeMO Tool. (Lanz 2010; Lanz et al. 2010.)

Pro-FMA Extended

The Pro-FMA Extended (Professional Feature Modelling and Analysis Tool Extended), developed by Garcia et al. (2011) is used to define the product requirements from the product model given in VRML or X3D format. It is an extended version of Pro-FMA (Garcia et al. 2010), which focuses on analysis of the product features, the re-creation of any lost geometric features and the addition of extra information needed for other process steps. Pro-FMA Extended is dedicated to both the recognition and analysis of the features, and the creation of the pre-process plan (see Figure 55). The pre-process plan is a generic recipe for how to manufacture a part or a product. Each feature contains its characteristics: shape, type, geometric dimensions, material and tolerance. The pre-process plan can be created using these basic elements. Basically it is an ordered graph of generic activities referring to specific levels on the capability taxonomy stored in the KB. (Garcia et al. 2011.)



Hinge1_Part1_A.x3d
(0) Cylinder - Pad
(1) Chamfer - Pad
(2) Chamfer - Pad
(3) Cylindrical - Hole
(4) Cylindrical - Hole
(5) Cylinder - Pad
(6) Chamfer - Pad
(7) Chamfer - Pad

Feature Info
Number = 5
Type = Pad
Shape = Cylinder
Sub-features = 2

FaceExtended = 8
[13-1] Circle
[18-0] Part of Chamfer
[19-0] Part of Chamfer
[16-0] Part of Cylinder
[17-0] Part of Cyl
[21-0] Part of Cha
[22-0] Part of Cha
[20-0] Circle

Face Extended Data
Type: Circle
Area = 572.555mm2
Radius = 13.5mm
Diameter = 27mm

Missing Data
☑ Tolerance + 0.3 - 0.0
☑ Material Aluminium
☑ Special finishing none

> Displaying 3D model
> Openning a X3D file format
>Program Starting

Material_Removing — Material_Removing — Material_Removing — Material_Removing — Material_Transporting — Material_Removing / Material_Removing — Material_Removing / Material_Removing — Material_Transporting

Information of process_11
Type: Material_Removing
Position: Finishing
Feature type:
Resoruces:

● Cylinder
○ Box
○ Sheet Metal
○ Other
Diameter 40 mm
Height 60 mm

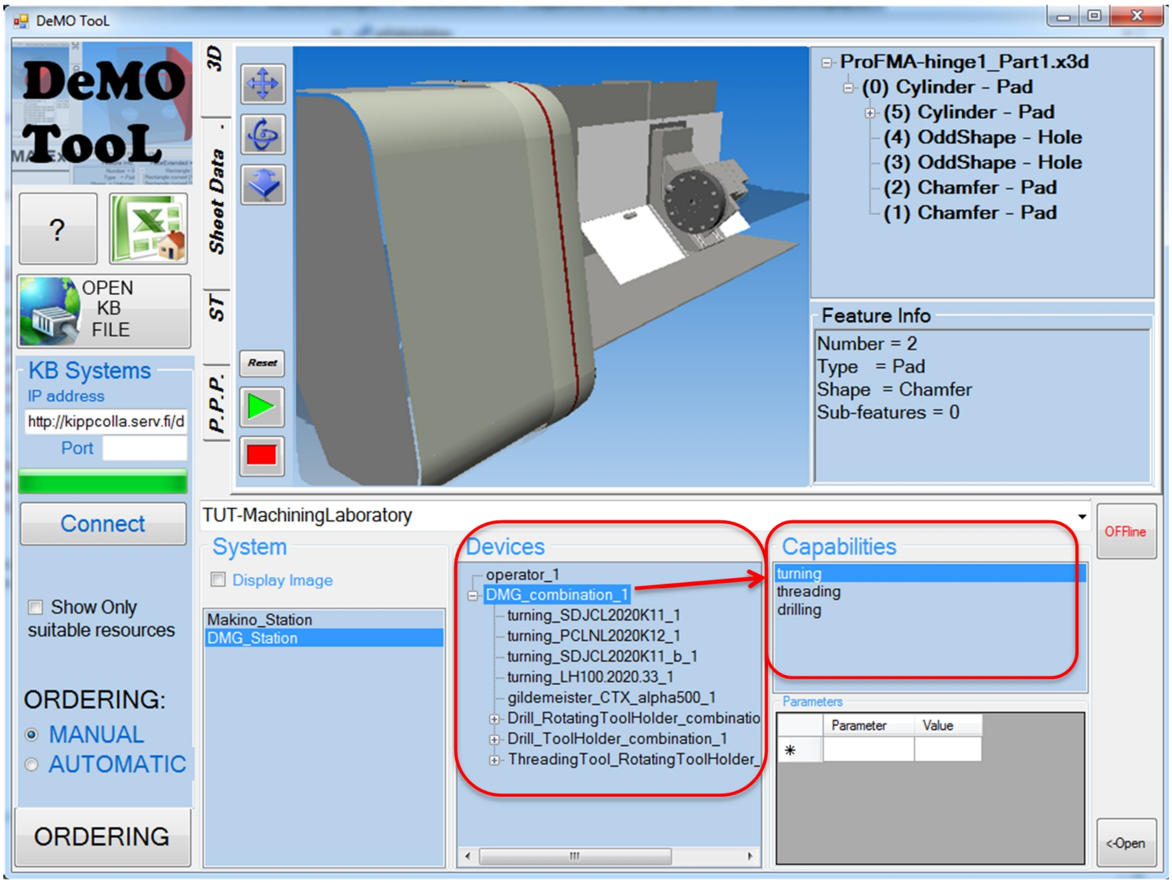
> Openning a X3D file format
>Program Starting

Figure 55. Pro-FMA Extended – Feature analysis and creation of the pre-process plan, modified from (Garcia et al. 2011).

DeMO Tool

The DeMO Tool (Decision Making and Ordering Tool) is meant for sending orders to the holonic manufacturing system and for viewing and analyzing the virtual manufacturing results. The part-ordering feature in the DeMO Tool contains controls for ordering product or individual feature manufacturing from the factory floor. The user can either let the holonic framework automatically route the product to a resource that has the required capability or choose the desired resource for manufacturing the product manually. The feature and resource information is obtained from the Knowledge Base. The order is sent to the factory floor through web services. With the DeMO Tool, it is also possible to run pre-created simulations of the processes and view the statistics, in order to evaluate and validate different production scenarios before placing the orders.

The different manufacturing stations which exist on the factory floor and are defined in the KB can be viewed through the DeMO tool. When selecting one station, this tool shows the individual machines and tools comprising the station. It also shows the capabilities and combined capabilities of the selected resources. Figure 56 shows the DeMO Tool GUI for viewing the resources. The red rectangles and the arrow show how the resources and their capability information are illustrated using this tool. In Figure 57, the lower-level device combinations and their capabilities are displayed. This resource information, originally developed by the Capability Editor, is extracted from the KB to the DeMO Tool. Based on the capability associations defined in the capability model, the DeMO tool is able to determine the combined capabilities of the combinations of individual devices having simple capabilities. This proves that the resource and capability information presented with the formal ontology can be utilized and understood by multiple different tools.



- ProFMA-hinge1_Part1.x3d
  - (0) Cylinder - Pad
    - (5) Cylinder - Pad
    - (4) OddShape - Hole
    - (3) OddShape - Hole
    - (2) Chamfer - Pad
    - (1) Chamfer - Pad

Number = 2
Type = Pad
Shape = Chamfer
Sub-features = 0

http://kippcolla.serv.fi/d

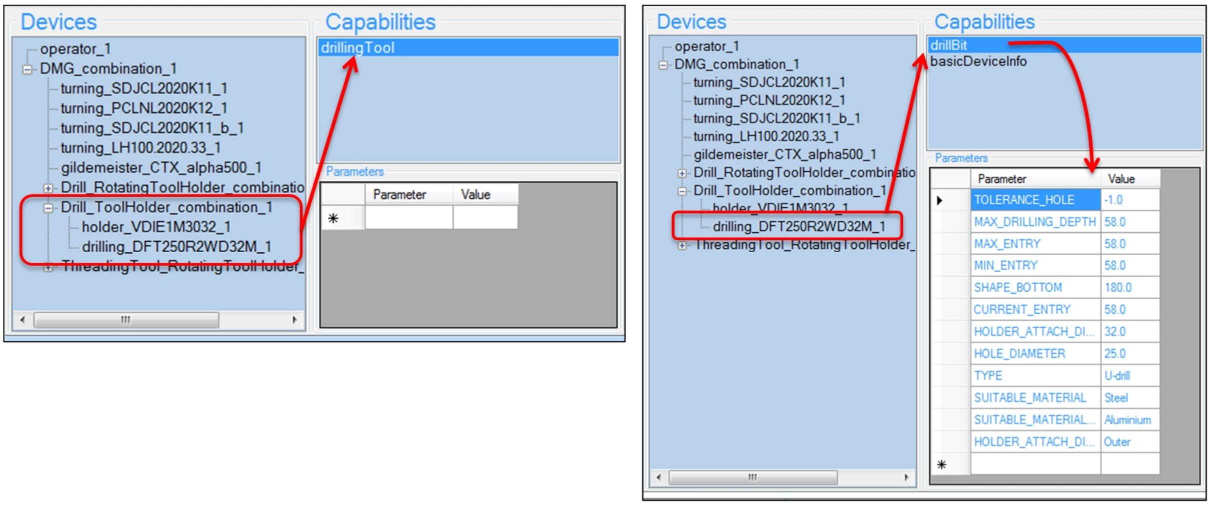
TUT-MachiningLaboratory

Makino_Station
DMG_Station

- operator_1
- DMG_combination_1
  - turning_SDJCL2020K11_1
  - turning_PCLNL2020K12_1
  - turning_SDJCL2020K11_b_1
  - turning_LH100.2020.33_1
  - gildemeister_CTX_alpha500_1
  - Drill_RotatingToolHolder_combinatio
  - Drill_ToolHolder_combination_1
  - ThreadingTool_RotatingToolHolder_

turning
threading
drilling

Figure 56. DeMO Tool – Viewing the resources on the factory floor.



Parameter	Value
TOLERANCE_HOLE	-1.0
MAX_DRILLING_DEPTH	58.0
MAX_ENTRY	58.0
MIN_ENTRY	58.0
SHAPE_BOTTOM	180.0
CURRENT_ENTRY	58.0
HOLDER_ATTACH_DI...	32.0
HOLE_DIAMETER	25.0
TYPE	U-drill
SUITABLE_MATERIAL	Steel
SUITABLE_MATERIAL...	Aluminium
HOLDER_ATTACH_DI...	Outer

Figure 57. DeMO Tool – Viewing the capabilities of the device combinations and individual devices.

Resource KB and Machine DB

The resource KB includes the information related to the resources and their capabilities. It forms a library of resources, including the individual devices on the factory floor, as well as the device blueprints representing available resources provided by the system providers. The resource descriptions are instances in the ontology. As the resource entities change during their lifecycle and usage, the resource description need to be updated over time. The Machine DB is meant for storing the real-time information relating to the resource's behavior and measured values from the factory floor. This raw data is filtered and relevant lifecycle key values, such as Mean Time Between Failure (MTBF), Mean Time to Repair (MTTR), maintenance costs and time, and operating time will be calculated and saved in the resource KB, as discussed in Chapter 4.3.1. This information can be used in the planning process for re-use and adaptation. The capability information of the individual resources is also updated based on the measured values from the factory floor.

Rule-base

The rule-base is developed as a store for the rules used in the capability matching as discussed in Chapter 4.4.4, in which a detailed description of the rules were given. The rules include domain expert rules, combined capability rules and adaptation rules (shown in pseudo code in Appendix 5). An extensive Python framework has been developed for writing the rules and for retrieving the information relating to the product requirements and resource capabilities from the Knowledge Base. The rules are saved as links to the ontology under the correct rule class, and refer to the capability taxonomy as discussed in Chapter 4.4.5, but the rules themselves are saved in a separate location, from where they can be called upon when needed. The implementation of the rule-base is an on-going project of another researcher. At this stage, only some of the rules, mainly those under the categories of "detailed capability definition" and "detailed capability matching" have been implemented, in order to validate the concept and to test the capability-matching.

UI/Control holon

The UI/Control holon manages the process flow and the distribution of the tasks to each manufacturing or assembly station. The UI/Control holon is the interface through which the other tools can communicate with the holonic manufacturing system. Through the Control Holon, users can pass requests to the holarchy. These requests are formally defined and documented. The Control

Holon has an XML-RPC interface that any UI can use to control the holon. The Control Holon can communicate with the resource UIs, and therefore it knows the status of each resource and can distribute the orders to the resources based on their capabilities and availability. (Järvenpää et al. 2011b)

5.2. Dynamic operation environment – adaptation by reaction

In a dynamic environment, the scheduling and dispatching of orders cannot be approached in a static way. The control of the internal material flow in the production system should also be managed in a dynamic way. In this case study, these dynamics are achieved through holonic self-organization. As discussed earlier, the holonic operating environment represents a dynamic system, where the adaptation takes place "online" by reaction.

The dynamic operation environment was developed in order to demonstrate the results of the KIPPcolla and CSM-hotel projects. The environment consists of the hardware in the TUT machining laboratory and the modular ICT-architecture discussed in the previous chapter. The build environment utilizes a holonic manufacturing paradigm and integrates existing technologies resulting from different projects into one operational environment.

The main characteristic of the developed holonic system is that the status of the production system and the desired goal (defined as order connected to product model) are known, but the steps for reaching the goal, in this case the routing of the parts on the factory floor, are not predefined. The holonic system adheres to a service-oriented architecture (SOA), where the resources provide services through their capabilities. When an order enters the holonic framework, the system will only search for those resources which can by themselves, or with some other resource, satisfy the requested service. The holons will then negotiate to determine the best resource for the given situation, or the part is directed to the first available resource combination that has the capabilities needed to produce the part or a specific feature. The holon implementation is based on a peer-to-peer network, where the holons are able to join and leave the holarchies freely and other holons can see which ones are currently online, i.e. which capabilities are available in the system. (Järvenpää et al. 2012a.)

The next chapter introduces the hardware of the dynamic operation environment. Following that, the procedures and activities arising during the case scenario will be discussed.

5.2.1. Hardware in the dynamic operation environment

The hardware part of the environment consists of several manufacturing resources and work pieces as physical manufacturing entities, the real parts (see Figure 58). Each of them has their corresponding computer models and simulation environments as their digital and virtual parts. Each machine on the factory floor also has its holon representation, which presents that specific machine in a digital world and retrieves the capability information from the Knowledge Base.

The physical resources in the research environment, offering different manufacturing capabilities, are:

- Machine tools (a lathe and a machining centre) for machining operations;
- Robots for material handling and robotized machining operations;

- Laser devices for marking and surface treatment;
- An automated storage for storing blank parts and finished work pieces;
- A punch press, existing only virtually, for the punching of sheet-metal parts.

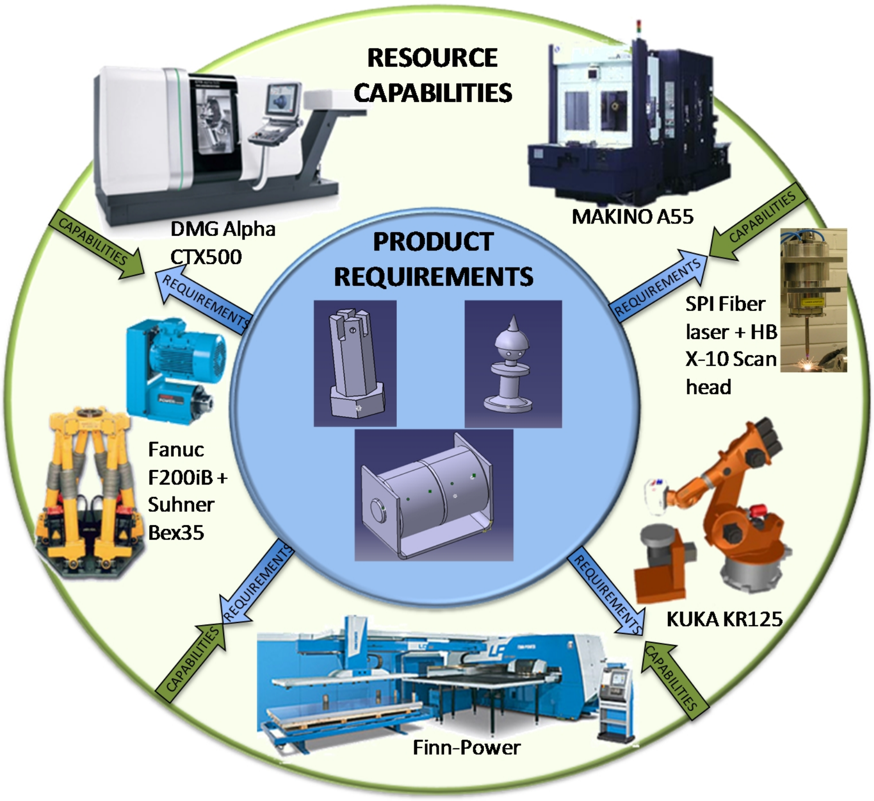


Figure 58. Resources on the factory floor and manufactured products (Järvenpää et al. 2012a).

The work pieces are fairly simple, being cuboid, cylindrical, or flat in shape. They have several features with parameters that can be altered, such as part dimensions (width, length, and depth), number of holes, corner radiuses and chamfers, sheet thickness and the material and tolerance requirements of the finished products. These features determine the product requirements for which suitable capabilities need to be found.

5.2.2. Process flow during the case scenario

This section will explain, step by step, the actions taken when a new product enters production in the dynamic operation environment. Figure 54 gave an overall view of the information flows in the operation environment. Figure 59 presents the simplified flow of the activities and reasoning process in the environment. This process description is divided into activities on the digital, virtual and real levels. The digital level refers to digital information saved in the ontology, the virtual level, on the other hand, means the simulation world. The real level refers to the actual, physical, production environment.

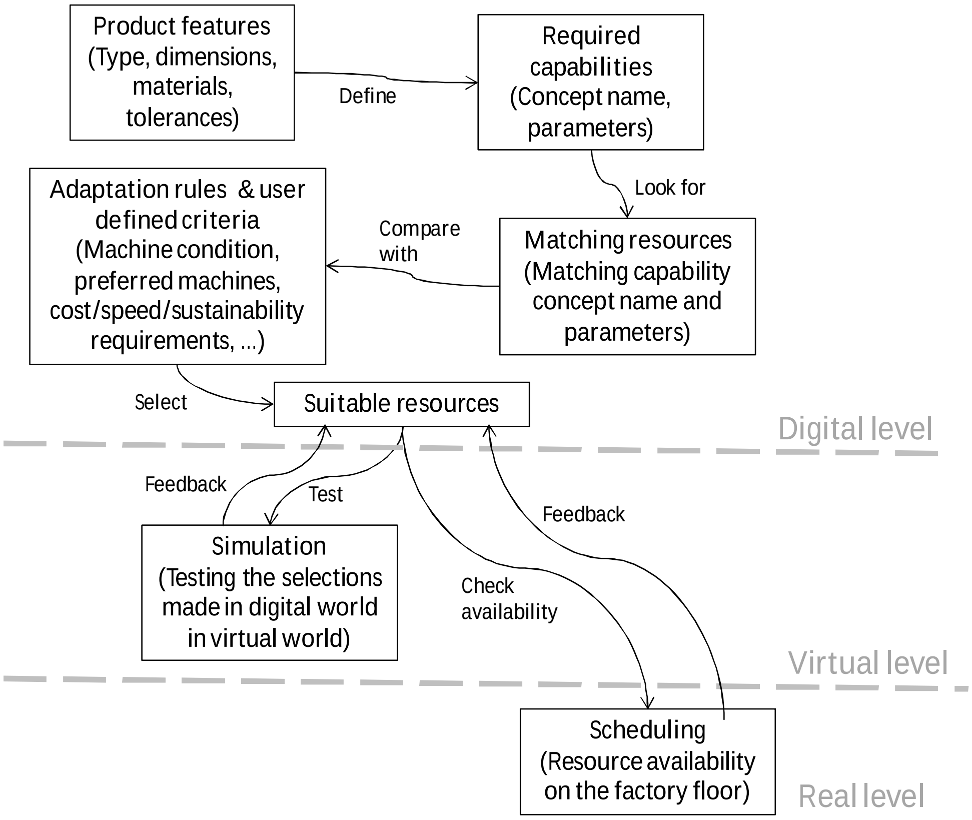

Product features (Type, dimensions, materials, tolerances)
Define
Required capabilities (Concept name, parameters)
Look for
Adaptation rules & user defined criteria (Machine condition, preferred machines, cost/speed/sustainability requirements, ...)
Compare with
Matching resources (Matching capability concept name and parameters)
Select
Suitable resources
Digital level
Feedback
Test
Feedback
Simulation (Testing the selections made in digital world in virtual world)
Check availability
Virtual level
Scheduling (Resource availability on the factory floor)
Real level

Figure 59. Process flow in the holonic manufacturing system framework (Järvenpää et al. 2011b).

Table 30 presents the actions taken during the case scenario and their level of execution. In the following text, each of these activities will be discussed in detail. A case product, shown in Figure 60, is used as an example to illustrate the case scenario.

Table 30. Activities in the case scenario.

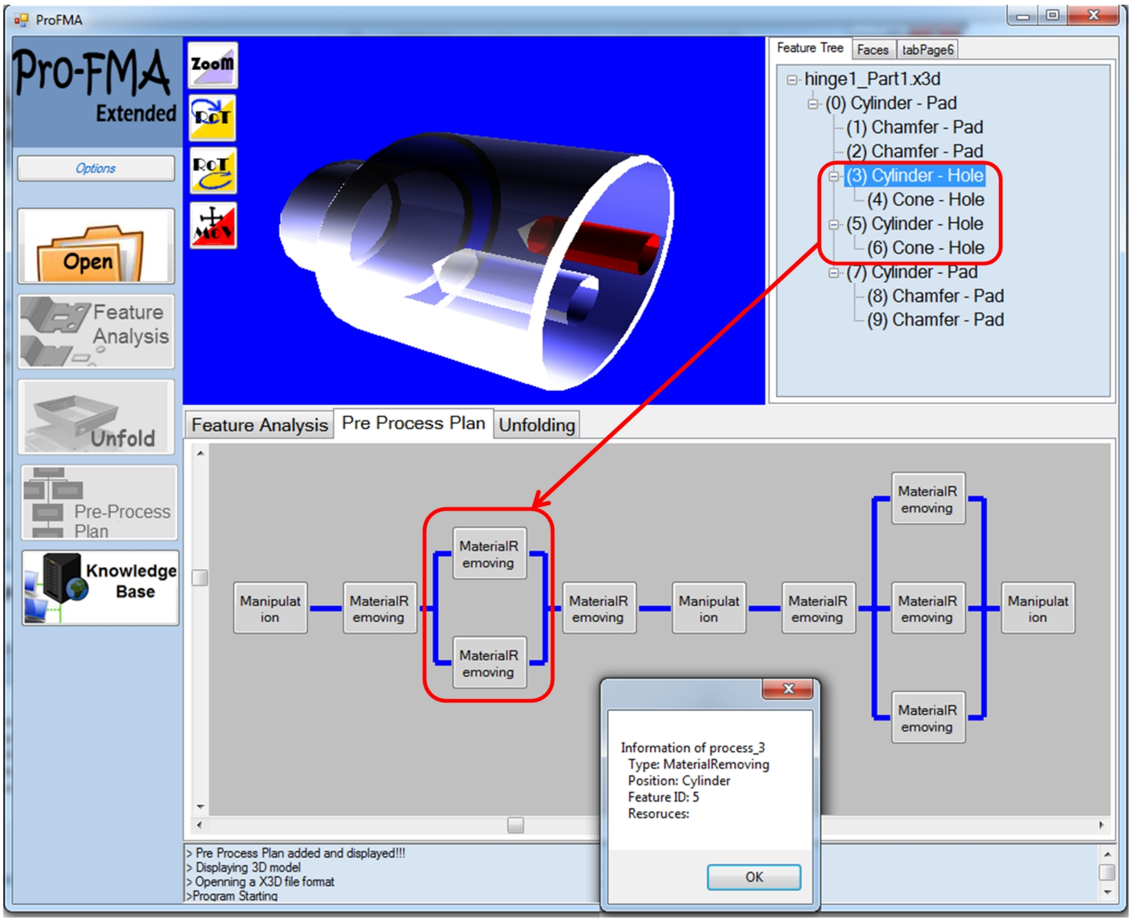
Levels	Activities
Digital level	1. Definition of the product features 2. Definition of the pre-process plan 3. Matching the capability requirements and resources 4. Applying the adaptation rules and user-given criteria for the resource selection
Virtual level	5. Test manufacturing by simulation
Real level	6. Checking the availability of resources and routing the order 7. Collecting the history information

1. Definition of the product features

The product model is first sent in VRML or X3D format to the Pro-FMA Extended software, which will recognize the features that need to be manufactured. It recognizes the shape, type and dimensions of the feature. The user can manually enter the material and tolerance information, as well as special finishing requirements for the features.

2. Definition of the pre-process plan

Based on the recognized features, the Pro-FMA Extended defines a pre-process plan for the manufacture of the part/product. The pre-process plan is created based on the rules embedded in the software (Garcia et al. 2011). This is created without any knowledge of the currently available resources on the factory floor. Therefore, instead of defining that, e.g. a “milling” or “turning” capability is needed, only very high-level generic capability requirements, e.g. “material removing”, are defined in the pre-process plan. This allows multiple machine options for manufacturing a certain feature, and thus allows the product to be dynamically routed to, for instance, the first available machine. The pre-process plan is then sent to the Knowledge Base, where the steps of the pre-process plan are linked with the capability taxonomy. Figure 60 shows the pre-process plan of the product used in this case scenario. The two holes are illustrated as the example features and associated pre-process plan steps.



Information of process_3
Type: MaterialRemoving
Position: Cylinder
Feature ID: 5
Resoruces:

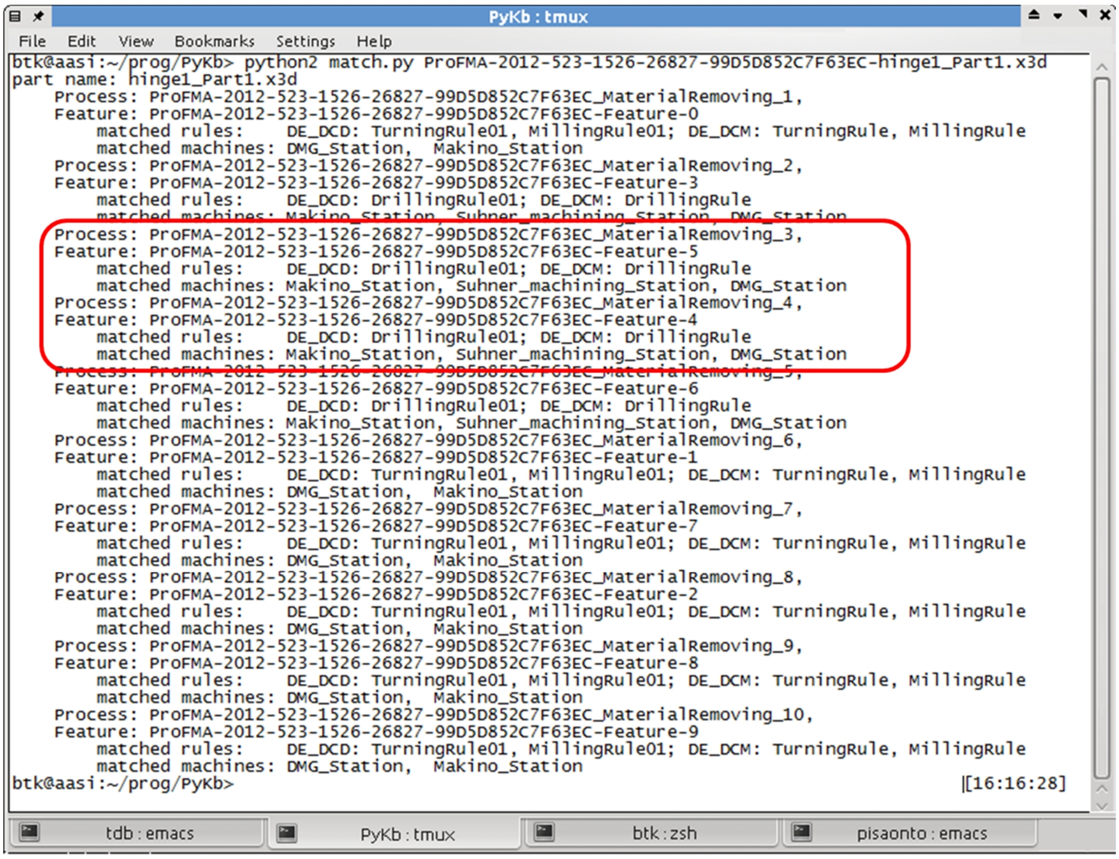
```
> Pre Process Plan added and displayed!!!
> Displaying 3D model
> Openning a X3D file format
>Program Starting
```

Figure 60. Pre-process plan of the case product.

The pre-process plan, together with the feature recognition data, defines the capability requirements for the manufacture of the part/product in the following way. The generic process names used in the pre-process plan refer to certain capabilities in the capability taxonomy. Each step in the graph represents one generic capability requirement. The parameters relating to the product features, e.g. type, size and material remain on the product side of the ontology. These parameters determine the detailed capability requirements.

3. Matching the capability requirements and resources

The matching of the required capabilities and suitable resources is done based on the capability taxonomy and the rule-base. Both the pre-process plan and the resource-specific capabilities have a reference to the capability taxonomy, allowing high-level capability mapping. The detailed matching is done based on the rules in the rule-base. The capability matching holon (i.e. a program that uses the capability matching rules) obtains the capabilities of each device taking part in the device combinations from the KB. Based on the rules, it uses the capability information and compares the capability parameters to the product requirements presented as the pre-process plan and its related feature information. This capability matching procedure was discussed in detail in Chapter 4.4. In Figure 61, the actual results from the capability matching program are shown. The red rectangle indicates the drilling processes from Figure 60.



```
btk@aasi:~/prog/PyKb> python2 match.py ProFMA-2012-523-1526-26827-99D5D852C7F63EC-hinge1_Part1.x3d
part name: hinge1_Part1.x3d
    Process: ProFMA-2012-523-1526-26827-99D5D852C7F63EC_MaterialRemoving_1,
    Feature: ProFMA-2012-523-1526-26827-99D5D852C7F63EC-Feature-0
        matched rules:    DE_DCD: TurningRule01, MillingRule01; DE_DCM: TurningRule, MillingRule
        matched machines: DMG_Station,  Makino_Station
    Process: ProFMA-2012-523-1526-26827-99D5D852C7F63EC_MaterialRemoving_2,
    Feature: ProFMA-2012-523-1526-26827-99D5D852C7F63EC-Feature-3
        matched rules:    DE_DCD: DrillingRule01; DE_DCM: DrillingRule
        matched machines: Makino_Station, Suhner_machining_Station, DMG_Station
    Process: ProFMA-2012-523-1526-26827-99D5D852C7F63EC_MaterialRemoving_3,
    Feature: ProFMA-2012-523-1526-26827-99D5D852C7F63EC-Feature-5
        matched rules:    DE_DCD: DrillingRule01; DE_DCM: DrillingRule
        matched machines: Makino_Station, Suhner_machining_Station, DMG_Station
    Process: ProFMA-2012-523-1526-26827-99D5D852C7F63EC_MaterialRemoving_4,
    Feature: ProFMA-2012-523-1526-26827-99D5D852C7F63EC-Feature-4
        matched rules:    DE_DCD: DrillingRule01; DE_DCM: DrillingRule
        matched machines: Makino_Station, Suhner_machining_Station, DMG_Station
    Process: ProFMA-2012-523-1526-26827-99D5D852C7F63EC_MaterialRemoving_5,
    Feature: ProFMA-2012-523-1526-26827-99D5D852C7F63EC-Feature-6
        matched rules:    DE_DCD: DrillingRule01; DE_DCM: DrillingRule
        matched machines: Makino_Station, Suhner_machining_Station, DMG_Station
    Process: ProFMA-2012-523-1526-26827-99D5D852C7F63EC_MaterialRemoving_6,
    Feature: ProFMA-2012-523-1526-26827-99D5D852C7F63EC-Feature-1
        matched rules:    DE_DCD: TurningRule01, MillingRule01; DE_DCM: TurningRule, MillingRule
        matched machines: DMG_Station,  Makino_Station
    Process: ProFMA-2012-523-1526-26827-99D5D852C7F63EC_MaterialRemoving_7,
    Feature: ProFMA-2012-523-1526-26827-99D5D852C7F63EC-Feature-7
        matched rules:    DE_DCD: TurningRule01, MillingRule01; DE_DCM: TurningRule, MillingRule
        matched machines: DMG_Station,  Makino_Station
    Process: ProFMA-2012-523-1526-26827-99D5D852C7F63EC_MaterialRemoving_8,
    Feature: ProFMA-2012-523-1526-26827-99D5D852C7F63EC-Feature-2
        matched rules:    DE_DCD: TurningRule01, MillingRule01; DE_DCM: TurningRule, MillingRule
        matched machines: DMG_Station,  Makino_Station
    Process: ProFMA-2012-523-1526-26827-99D5D852C7F63EC_MaterialRemoving_9,
    Feature: ProFMA-2012-523-1526-26827-99D5D852C7F63EC-Feature-8
        matched rules:    DE_DCD: TurningRule01, MillingRule01; DE_DCM: TurningRule, MillingRule
        matched machines: DMG_Station,  Makino_Station
    Process: ProFMA-2012-523-1526-26827-99D5D852C7F63EC_MaterialRemoving_10,
    Feature: ProFMA-2012-523-1526-26827-99D5D852C7F63EC-Feature-9
        matched rules:    DE_DCD: TurningRule01, MillingRule01; DE_DCM: TurningRule, MillingRule
        matched machines: DMG_Station,  Makino_Station
btk@aasi:~/prog/PyKb>                                                                    |[16:16:28]
```

Figure 61. View from the capability matching program.

During the case scenario the matching is performed as follows:

1) First the domain expert rules for detailed capability definition are used to define what capabilities are actually useable for that specific "MaterialRemoving" process. The rules having reference to "MaterialRemoving" in the capability taxonomy are checked one by one until a match is found (i.e. the rule returns the value TRUE). These rules are indicated in the figure with the prefix "DE_DCD". For example, in the case of the hole feature shown in Figure 60, these rules will indicate that a "drilling" capability is needed. In this way, all the resources which don't have the "drilling" capability are filtered out.
2) After the required capabilities have been defined on the detail level, then the domain expert rules for detailed capability matching are used to define if the resources which really match with the requirements on the parameter level actually exist. The selected rules are indicated in the figure with the prefix "DE_DCM". These rules will compare the feature properties with the capability parameters. For example, in the case of the hole feature, the detailed capability matching rule which is specified with the taxonomy level "Drilling" will be triggered. This rule compares all the resources that have the "drilling" capability to the required features at the parameter level and returns all the matching resources.

Figure 61 shows the applied rules and the resources which were found to match. Table 31 shows those rules that were applied in the matching phases illustrated with the red rectangle. All the other rules can be found in Appendix 5.

Table 31. Rules applied for the hole features of the case product.

Rule acronym	Rule

DE_DCD
DrillingRule01
	IF feature.hasCapabilityTaxonomy(“*MaterialRemoving*”) **AND**
	feature.hasFeatureType (“hole”) **AND**
	feature.hasFeatureType(“cylindrical”) **AND**
	feature.getParameter(“diameter”) **<= 40.0**
	THEN
	RETURN [Capabilities.find(“*drilling*”)]

DE_DCM DrillingRule	**IF** feature.getParam(“diameter”) = *drillBit*.getParam(“hole_diameter”) **AND** feature.getParam(“depth”) <= *drillBit*.getParam(“max_drilling_depth”) **AND** feature.getParam(“bottomShape”) = *drillBit*.getParam(“shape_bottom”) **AND** feature.getMaterial() **IN** *drillBit*.suitableMaterials() **AND** product.getInitialProduct().size().isInside(*fixturing*.minItemsize()- *fixturing*.maxItemSize()) **THEN** **RETURN TRUE**

As can be seen from Figure 61, three resource options for the drilling process steps can be found: the Suhner_machining_Station, the DMG_Station and the Makino_Station. For the other steps, only two resource options, the DMG_Station and the Makino_Station, are returned. This is because the Suhner_machining_Station is currently only equipped with the drilling tool, whereas the DMG_Station and Makino_Station have automatic tool changing systems and multiple tools that they can use for drilling, milling and turning.

4. Applying adaptation rules and user-given criteria for the resource selection

The DeMO Tool is used to place the order and send it to the factory floor, as well as to select the preferred resources and apply the user-defined criteria. First, the product to be ordered is selected. The analysed feature information related to that specific product is retrieved from the Knowledge Base. Based on the detailed capability matching done in the previous phase, the DeMO Tool can show the user the resources whose capabilities match the requirement. Should the user want to manually select the resource to be used, this view allows the user to evaluate the capabilities of the existing resources and to send the order to a specific machine. For example, in this case scenario the DeMO tool would suggest both the DMG_Station, as shown in Figure 56, and the Makino_station. Otherwise the holonic framework makes the decision automatically, taking into account the availability, adaptation rules and the user-given criteria. The user-given criteria can relate to, for example, sustainable performance metrics, such as energy consumption, waste generation and costs, or machine condition, speed, and so on.

5. Test manufacturing by simulation

With the DeMO tool, it is possible to run pre-created simulations of the processes and view the statistics before placing the orders. Because it is very difficult to determine some capabilities accurately on a digital level, such as the workspace and reach of a device combination, virtual simulations may be required to validate the feasibility of the found capabilities.

6. Checking the availability of resources and routing the order

When the order is placed by the user, the UI/Control holon will check the availability and status of the suitable resources on the factory floor by negotiating with the machine UIs. The information about the machine status is then communicated back to the DeMO Tool. Next the order is routed to the first available and suitable manufacturing resource, taking into consideration the adaptation rules and the user-given criteria.

7. Collecting the history information
The machine database saves the resource behavior log information, including the status of the resources, completed orders, measured operational values, and so on. This information may later be used for decision making.

5.2.3. Discussion of the demonstration

The resource description, the capability model, and the capability matching framework comprising the capability taxonomy and the rules, all play a crucial role in the demonstration, enabling the resource holons to advertise their capabilities, orders to express their required capabilities, and finally a match to be made between these factors. At present, the rule-base is not yet fully implemented as part of the dynamic operation environment so some simplifications still exist in the current status of the demonstration. Only a few rules have been implemented, so the orders that can be sent to the system are limited.

Descriptions of all the involved hardware resources were created for the demonstration. The capabilities of these resources were included in the instantiated capability model. Furthermore, rules were created to enable the matching of product requirements and resource capabilities in the context of the use cases in question. These rules can be found in Appendix 5. The fact that it was possible to create these rules which compare the product and the resource information saved in the ontology, and provide feasible matches, proves that the resource description and capability model are expressive enough for such use cases. In addition, even though the rule-base is still under construction, the implemented rules prove that automatic matching is possible.

The demonstration shows that the capability taxonomy and the rules help to automatically filter the information and allow suitable solutions to be found from the vast amount of input information. Based on the resource and capability descriptions, it was possible to filter out those device combinations which didn’t provide the suitable capabilities. In large factories with thousands of resources, for example, this automatic filtering can provide remarkable savings in time.

5.3. Microfactory system – adaptation by planning

The second case study aims to describe how the developed capability-based adaptation methodology is applied and how the capability matching rules are used in a static adaptation context. In this thesis, the microfactory system represents a static system where the adaptation takes place “offline”, based on human-centric planning. In the microfactory environment, modularization and standardized interfaces are used as enablers for the adaptation. In the case study presented in this chapter, the process plan is predefined before sending the order and starting the production. Therefore, the reasoning can start with a more detailed process plan than in the previous case study. The main task in the following case study is to evaluate if the existing TUT-microfactory system has the capabilities to cope with the requirements set by the product assembly. First, the existing system capabilities are presented, followed by a definition of the product requirements. After that, the matching of the product requirements against the system capabilities based on the capability matching rules will be explained in detail. Finally, a few case scenarios of changing requirements, their compatibility with the existing system and the adaptation actions required for the system will be discussed in order to demonstrate the use of the compatibility domain approach.

5.3.1. Definition of system capabilities

The existing system consists of a TUT-microfactory module, a cartesian manipulator, a screwdriver unit, a feeding system, a belt conveyor and a machine vision system with 2 camera units, as shown in Figure 62.

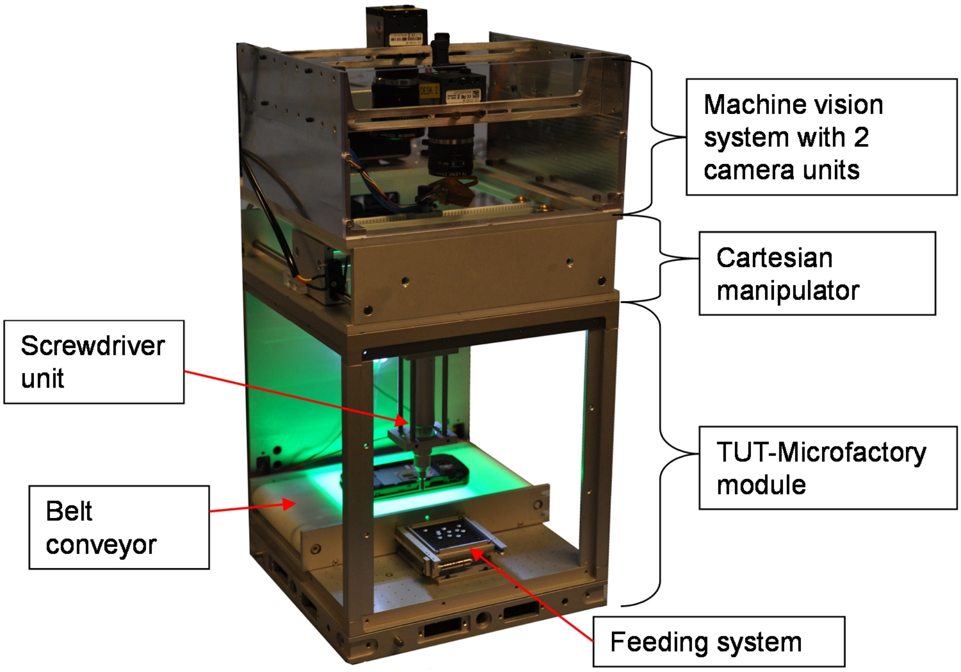
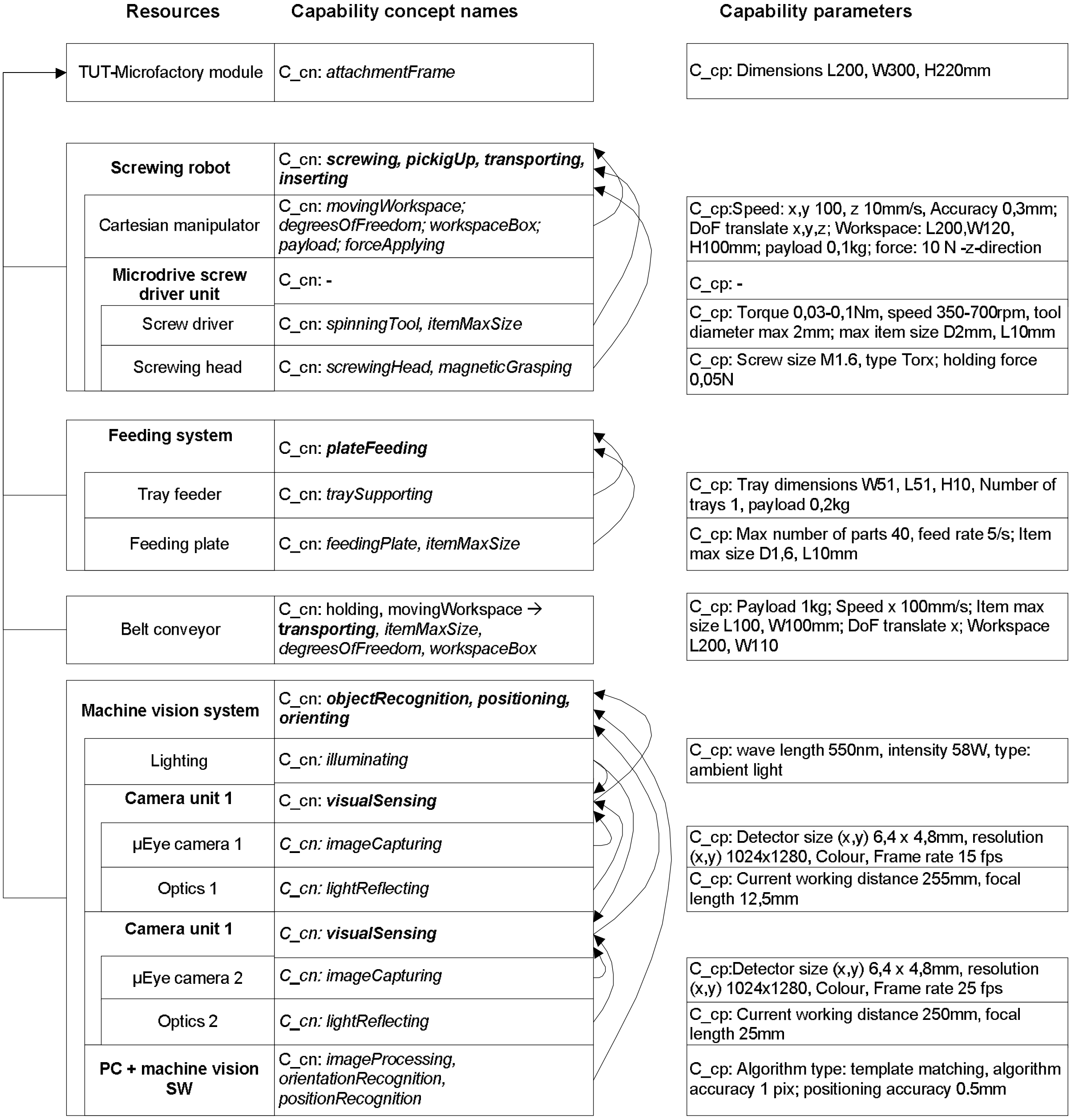


Figure 62. Existing TUT-microfactory system.

Figure 63 shows the capabilities of the existing microfactory system. Only the capabilities and capability parameters relevant to this case are displayed in the figure in order to maintain its readability. All the capabilities are saved to the ontology through the Capability Editor tool. The devices are grouped in their natural combinations, e.g. the camera unit consists of camera and optics, whereas the machine vision system consists of the camera unit, a PC and ambient lighting. The same grouping is also used for the description of the system in the ontology. The arrows indicate to which combined capability the simple capabilities of the devices contribute.



Resources	Capability concept names	Capability parameters
TUT-Microfactory module	C_cn: *attachmentFrame*	C_cp: Dimensions L200, W300, H220mm
Screwing robot	C_cn: ***screwing, pickigUp, transporting, inserting***	
Cartesian manipulator	C_cn: *movingWorkspace; degreesOfFreedom; workspaceBox; payload; forceApplying*	C_cp:Speed: x,y 100, z 10mm/s, Accuracy 0,3mm; DoF translate x,y,z; Workspace: L200,W120, H100mm; payload 0,1kg; force: 10 N -z-direction
Microdrive screw driver unit	C_cn: -	C_cp: -
Screw driver	C_cn: *spinningTool, itemMaxSize*	C_cp: Torque 0,03-0,1Nm, speed 350-700rpm, tool diameter max 2mm; max item size D2mm, L10mm
Screwing head	C_cn: *screwingHead, magneticGrasping*	C_cp: Screw size M1.6, type Torx; holding force 0,05N
Feeding system	C_cn: ***plateFeeding***	
Tray feeder	C_cn: *traySupporting*	C_cp: Tray dimensions W51, L51, H10, Number of trays 1, payload 0,2kg
Feeding plate	C_cn: *feedingPlate, itemMaxSize*	C_cp: Max number of parts 40, feed rate 5/s; Item max size D1,6, L10mm
Belt conveyor	C_cn: holding, movingWorkspace → ***transporting***, *itemMaxSize, degreesOfFreedom, workspaceBox*	C_cp: Payload 1kg; Speed x 100mm/s; Item max size L100, W100mm; DoF translate x; Workspace L200, W110
Machine vision system	C_cn: ***objectRecognition, positioning, orienting***	
Lighting	C_cn*: illuminating*	C_cp: wave length 550nm, intensity 58W, type: ambient light
Camera unit 1	C_cn: ***visualSensing***	
µEye camera 1	*C_cn: imageCapturing*	C_cp: Detector size (x,y) 6,4 x 4,8mm, resolution (x,y) 1024x1280, Colour, Frame rate 15 fps
Optics 1	*C_cn: lightReflecting*	C_cp: Current working distance 255mm, focal length 12,5mm
Camera unit 1	*C_cn:* ***visualSensing***	
µEye camera 2	*C_cn: imageCapturing*	C_cp:Detector size (x,y) 6,4 x 4,8mm, resolution (x,y) 1024x1280, Colour, Frame rate 25 fps
Optics 2	*C_cn: lightReflecting*	C_cp: Current working distance 250mm, focal length 25mm
PC + machine vision SW	C_cn: *imageProcessing, orientationRecognition, positionRecognition*	C_cp: Algorithm type: template matching, algorithm accuracy 1 pix; positioning accuracy 0.5mm

Figure 63. Capabilities of the existing TUT-microfactory system.

The system architecture of the TUT-microfactory concept places some constraints on the possible layout and configuration of the modules. In the TUT-microfactory concept, the microfactory frame is needed to contain the process module and auxiliary devices. The frame modules can be organized in different topologies. One process module can be attached on top of the frame module, the control module can be attached on top of the process module and auxiliary devices, feeders and other devices can be freely placed on the edges of the frame, or inside the frame if there is enough space. For the case study, one pre-condition is that the product should be produced with the TUT-microfactory. Therefore, the constraints set by the system architecture need to be considered in the capability matching.

5.3.2. Definition of the product requirements

The case product is a cell phone and the needed process is to attach four screws to the cell phone body, as shown in Figure 64. The product requirements are shown in the graph in Figure 65. On the left-hand side is the user-defined process plan and on the right-hand side are the product characteristics, which affect the required capabilities in each process phase. The steps in the process plan have a direct link to the capability taxonomy, which allows the mapping between the product requirements and existing system capabilities at the concept name level.

Additionally, the requirements, which are more project and user-preference related rather than product related, such as the required speed for transporting the phone and the desired type of feed, are shown in the figure. As discussed earlier, the selected system architecture, i.e. the TUT-microfactory architecture, places additional requirements and constraints on the required capabilities. For example, as the width of the TUT-microfactory module is known, then the transporting distance of the cell phone from one side of the module to the other is pre-defined. Some of these requirements are set by the selection of other devices in the system. For instance, the required fields of view of the camera units are not only determined by the product size, but also by the means of transportation and feed for the product and parts. In other words, the selection of some devices propagates some further new requirements. Examples of these kinds of requirements are included in the case example and are also shown in Figure 65.

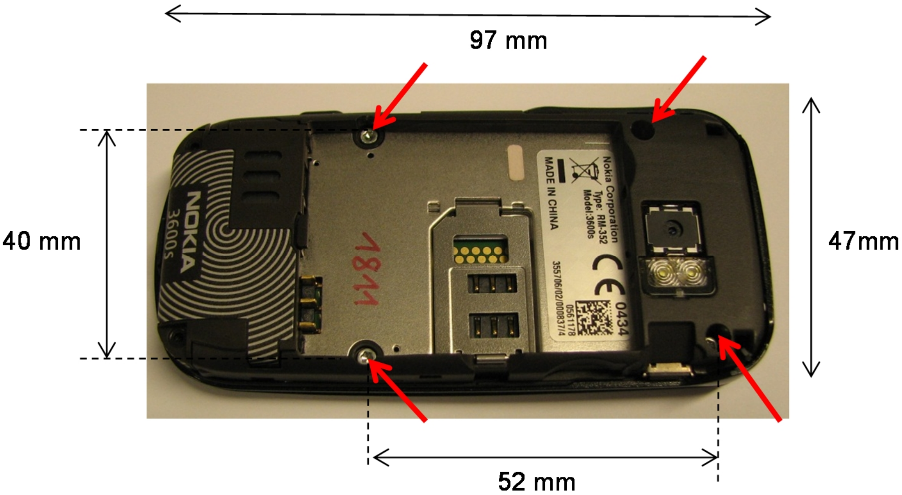
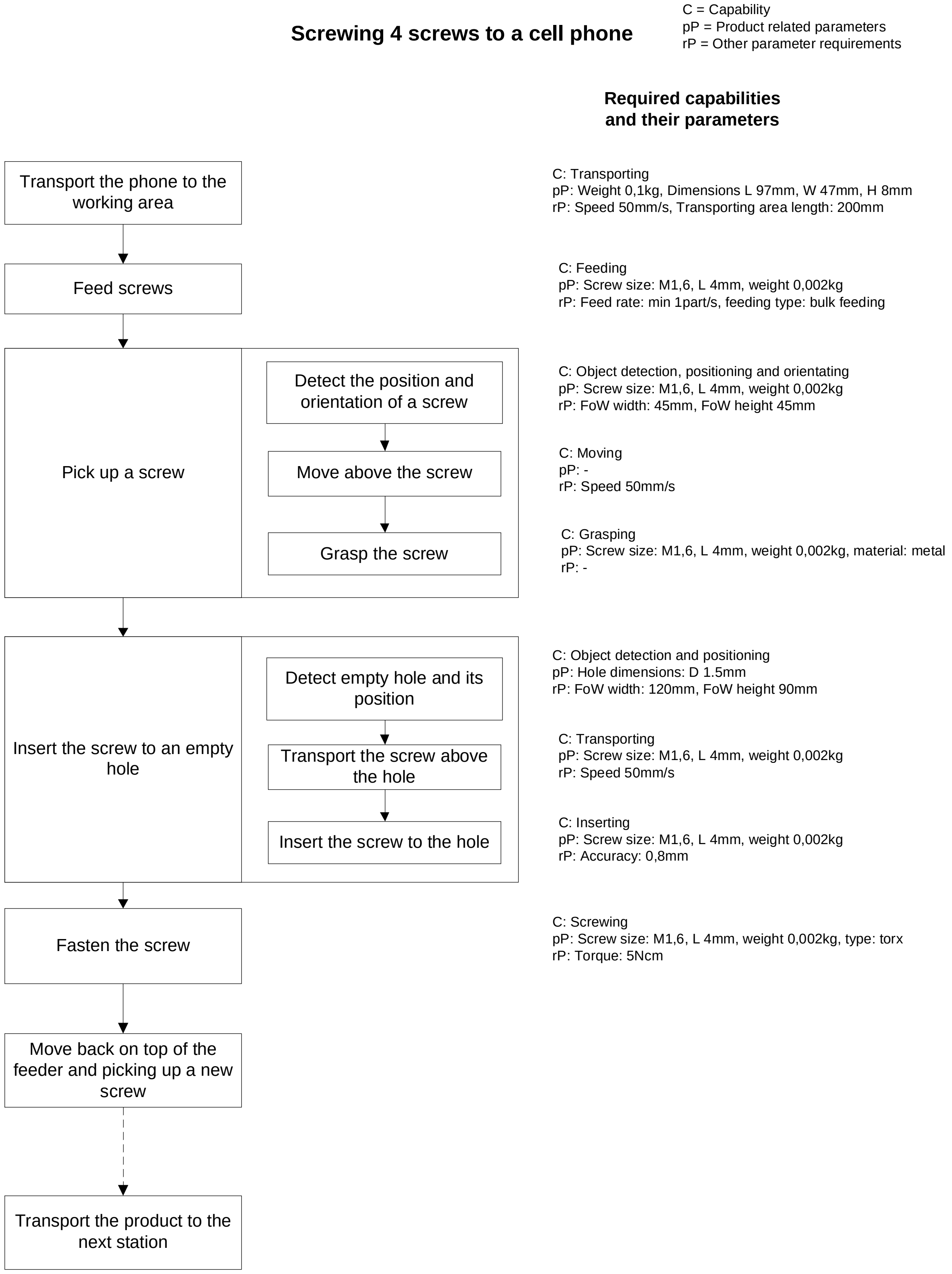


Figure 64. Case product – cell phone with four screws.


Screwing 4 screws to a cell phone
Transport the phone to the working area
Feed screws
Pick up a screw
Detect the position and orientation of a screw
Move above the screw
Grasp the screw
Insert the screw to an empty hole
Detect empty hole and its position
Transport the screw above the hole
Insert the screw to the hole
Fasten the screw
Move back on top of the feeder and picking up a new screw
Transport the product to the next station

C = Capability
pP = Product related parameters
rP = Other parameter requirements

Required capabilities and their parameters

C: Transporting
pP: Weight 0,1kg, Dimensions L 97mm, W 47mm, H 8mm
rP: Speed 50mm/s, Transporting area length: 200mm

C: Feeding
pP: Screw size: M1,6, L 4mm, weight 0,002kg
rP: Feed rate: min 1part/s, feeding type: bulk feeding

C: Object detection, positioning and orientating
pP: Screw size: M1,6, L 4mm, weight 0,002kg
rP: FoW width: 45mm, FoW height 45mm

C: Moving
pP: -
rP: Speed 50mm/s

C: Grasping
pP: Screw size: M1,6, L 4mm, weight 0,002kg, material: metal
rP: -

C: Object detection and positioning
pP: Hole dimensions: D 1.5mm
rP: FoW width: 120mm, FoW height 90mm

C: Transporting
pP: Screw size: M1,6, L 4mm, weight 0,002kg
rP: Speed 50mm/s

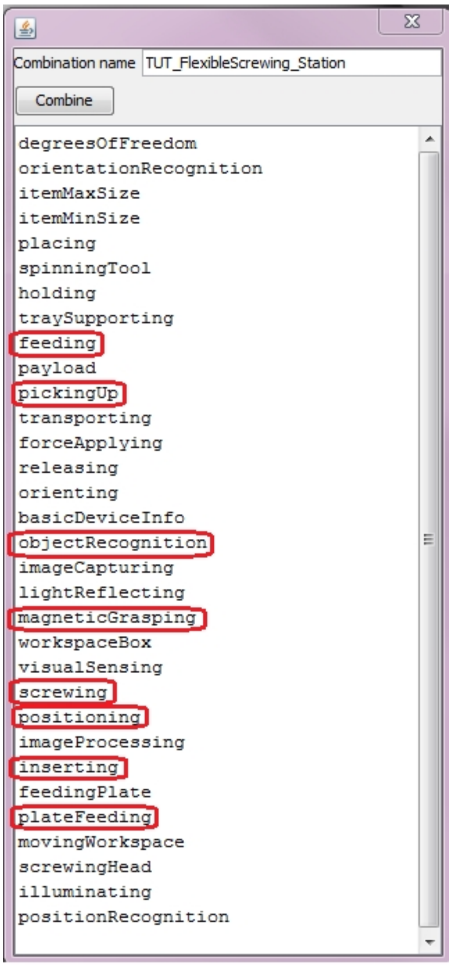
C: Inserting
pP: Screw size: M1,6, L 4mm, weight 0,002kg
rP: Accuracy: 0,8mm

C: Screwing
pP: Screw size: M1,6, L 4mm, weight 0,002kg, type: torx
rP: Torque: 5Ncm

Figure 65. Pre-defined process plan and related capability requirements.

5.3.3. Matching the product requirements against the system capabilities

Based on the description of the product requirements, the existing system capabilities, the capability taxonomy and the rules for the detailed capability matching, it is possible to reason out if the existing system has all the required capabilities needed to perform the screwing operations. Figure 66 shows all the capabilities that are assigned to the resources forming the current TUT-microfactory system for flexible screwing. The list also shows the combined capabilities that are formed from the simple ones. The list is created by the Capability Editor, which is able to combine the simple capabilities into combined ones based on the defined capability associations in the capability model. The combined capabilities that are relevant for the case product are marked with the red rectangles. The high-level capability mapping, which is done based on the capability taxonomy, shows that all the required capabilities at the concept name level can be found from the existing system. Next, the detailed capability matching needs to be performed based on the rules in the rule-base. In the following paragraphs, the reasoning procedures are explained step-by-step.



Combination name TUT_FlexibleScrewing_Station

```
degreesOfFreedom
orientationRecognition
itemMaxSize
itemMinSize
placing
spinningTool
holding
traySupporting
feeding
payload
pickingUp
transporting
forceApplying
releasing
orienting
basicDeviceInfo
objectRecognition
imageCapturing
lightReflecting
magneticGrasping
workspaceBox
visualSensing
screwing
positioning
imageProcessing
inserting
feedingPlate
plateFeeding
movingWorkspace
screwingHead
illuminating
positionRecognition
```

Figure 66. Capability Editor displaying all the capabilities of the resources forming the TUT-Flexible screwing station. The combined capabilities needed in the case are shown inside the red rectangles.

1) Transport the phone to the working area

The high-level capability mapping detects two devices (combinations) from the system which have the "transporting" capability. These are the belt conveyor and the screwing robot. The detailed matching first checks whether the product size and weight are suitable for the current devices, and secondly it checks if the workspace of the device and the speed of the transporting capability matches the requirement. The rules used for the matching are shown below.

RULE 1a: Is the product size suitable for the device?

```
conveyor = resource.hasCapability(“movingWorkspace”) AND (“holding”)

IF product.getParam(“weight”) <= holding.getParam(“payload”) AND
   product.getParam(“length”) <= conveyor.itemMaxSize.getParam(“length”) AND
   product.getParam(“width”) <= conveyor.itemMaxSize.getParam(“width”) AND
   product.getParam(“height”) <= conveyor.itemMaxSize.getParam(“height)
THEN
   RETURN TRUE
```

RULE 1b: Are the workspace and speed requirements met?

```
IF product.transportingArea(“length”) <= workspaceBox.getParam(“length”) AND
   product.transportingSpeed(“speed”) <= movingWorkspace.getParam(“speed_x”)
THEN
   RETURN TRUE
```

When these rules are applied to the presented case, i.e. filled with the parameter values shown in Figure 63 and Figure 64, it reveals that the belt conveyor capabilities match the requirements. Only the rules for the conveyor are shown here. However, similar reasoning with the screwing robot is shown later in this example, in RULES 3a and 3b. Those rules would immediately reveal that the screwing robot is not a suitable device combination for transporting the cell phone, because of the size, weight and material of the product.

2) Feed screws

Based on the high-level capability mapping, one device combination in the current system has the capability “*plateFeeding*”. This is the feeding system. The user has specified that the screws should be fed by a bulk feeding method in order to ease the manual handling of the screws. According to the capability taxonomy, the “*plateFeeding*” is a specialization of “*bulkFeeding*” and therefore fulfils the requirement. Because plate feeding is a method which doesn’t provide the parts in a certain position and orientation, the machine vision system, or another system providing “objectRecognition”, “positioning” and “orienting” capabilities is required to be able to detect the parts that can be picked up from the feeder. As shown in Figure 63 and Figure 66, these capabilities are also available in the system. The rules for detailed capability matching are now used to find out if the existing feeding system is able to feed the screws. First, it needs to be checked if the part size is suitable for the feeder (RULE 2a). Secondly, it is checked that the other requirements are fulfilled (RULES 2b and 2c).

RULE 2a: Is the screw size suitable for the feeder?

```
plate = resource.hasCapability(“feedingPlate”)

IF product.getParam(“diameter”) <= plate.itemMaxSize.getParam(“diameter”) AND
   product.getParam(“length”) <= plate.itemMaxSize.getParam(“length”) AND
THEN
   RETURN TRUE
```

The number of parts that can be poured into the feeder needs to be defined based on the payload of the feeder and the defined maximum amount of parts on the plate. The screws of the case product are stored in little plastic bags containing 40 screws. It would be convenient to pour the whole contents of the bag into the feeder. The rules that are needed to define whether that requirement can be satisfied are shown below.

RULE 2b: Payload of the feeder

```
Payload_of_plate_feeder [feeder + plate] = traySupporting.getParam(“payload”) –
(traySupporting.getParam(“number_of_trays”) * plate.basicDeviceInfo.getParam(“weight”))
```

RULE 2c: Can the feeder handle the required number of screws?

```
IF Desired_amount_of_products_in_feeder <= MIN(feedingPlate.getParam(“number_of_parts”),
   feederPayload/product.getParam(“weight”))
THEN
   RETURN TRUE
```

The matching shows that the current feeding system is able to feed the screws, and the requirement for the amount of screws in the feeder is just satisfied. However, the position and orientation of the screws also need to be detected in order to feed the parts in a specified position and orientation. The high-level mapping finds that the machine vision system consisting of two cameras has the capabilities of “object detection”, “positioning” and “orienting”. The physical arrangement of the machines, acquired from the virtual model, defines the position of the camera units in relation to the feeder and conveyor. Camera unit 1 is above the conveyor and Camera unit 2 is above the feeder. The working distances of the cameras are pre-defined based on the current installation.

The field of view (FoV) of the camera system is calculated based on the working distance of the camera, the detector size (CCD width x CCD length) and the focal length of the optics as shown in the formulas below.

$$FoV\ width = \frac{Working\ Distance * CCD\ width}{Focal\ Length} \tag{8}$$

$$FoV\ height = \frac{Working\ Distance * CCD\ height}{Focal\ Length} \tag{9}$$

RULE 2d: Field of view (workspace) of camera system

```
FoV_width = (lightReflecting.getParam(“current_working_distance”) *
imageCapturing.getParam(“detector_size_x”)) / lightReflecting.getParam(“focal_length”)

FoV_height = (lightReflecting.getParam(“current_working_distance”) *
imageCapturing.getParam(“detector_size_y”)) / lightReflecting.getParam(“focal_length”)
```

Based on the formulas (8) and (9) above, the field of view (W x H) of the camera systems are:

- Camera unit 1 FoV: 130 x 98 mm
- Camera unit 2 FoV: 60 x 48 mm

RULE 2e: Is the camera system field of view enough for the required application?

```
IF desired_view_height <= FoV_height AND
   Desired_view_width <= FoV_width
THEN
   RETURN TRUE
```

The desired field of view is defined after the feeding plate has been selected. Based on the plate size, the desired FoV is 45 x 45 mm. Camera unit 2 does have a bigger FoV, so it is suitable. The next thing that needs to be checked is whether the resolution of the camera is enough for the application. The minimum required detector resolution is calculated with the Nyquist principle based on the field of view and the smallest detectable feature, as shown in the formula (10). However, generally it is suggested that the resolution should be better than that suggested by the formula, possibly 2 to 5 times better than that.

$$Minimum\ detector\ resolution = 2 * \frac{Field\ of\ view}{Smallest\ feature} \tag{10}$$

RULE 2f: Does the camera system have enough resolution for detecting the screws?

```
IF product.getSmallestParam.size() >= 2 * FoV_width /
   imageCapturing.getParam(“x_resolution”) AND
   product.getSmallesParam.size() >= 2 * FoV_height /
   imageCapturing.getParam(“y_resolution”)
THEN
   RETURN TRUE
```

As the screws to be detected are 1.6 mm, the minimum detector resolution is 75 x 60 pixels, but preferably 375 x 300 pixels. The detailed capability matching shows that the camera resolution goes well beyond the required resolution.

3) Pick up a screw

The high-level mapping finds one device combination having the “pickingUp” capability. This is the screwing robot consisting of the cartesian manipulator and the screwdriver unit. The detailed capability matching needs to check if the screws can be grasped with the current device combination.

RULE 3a: Combined payload of robot + screw driver

```
robot = resource.hasCapability(“movingWorkspace”)
screwdriver = resource.hasCapability(“spinningTool”)
screwinghead = resource.hasCapability(“screwingHead”)

Combined_payload_of[robot + screwdriver] = MIN(robot.payload.getParam(“weight”) -
screwdriver.basicDeviceInfo.getParam(“weight”) –
screwinghead.basicDeviceInfo.getParam(“weight”)),
1/9,81 * magneticGrasping.getParam(“holding_force”))
```

RULE 3b: Can the screw be picked up by the device combination?

```
IF providedCapability = “magneticGrasping” AND
   product.getParam(“material”) = metal AND
   product.getParam(“weight”) <= Combined payload of [robot + screwdriver] AND
   product.getParam(“diameter”) <= screwdriver.maxItemSize.getParam(“diameter”) AND
   product.getParam(“length”) <= screwdriver.maxItemSize.getParam(“length”) AND
THEN
   RETURN TRUE
```

RULE 3c: Is the speed requirement met?

```
IF process.getParam(“speed_x”) <= movingWorkspace.getParam(“speed_x”) AND
   process.getParam(“speed_y) <= movingWorkspace.getParam(“speed_y”) AND
   process.getParam(“speed_z) <= movingWorkspace.getParam(“speed_z”)
THEN
   RETURN TRUE
```

Based on the detailed capability matching, the robot-screwdriver combination is able to pick up the screw and move at the desired speed.

4) Inserting the screws into an empty hole

Because the transportation of the cell phone on the belt conveyor doesn’t position and orient the product on the line, the positions of the holes are not known. Therefore, a method to detect the empty hole and its position has to be available. As discussed earlier in the ‘feeding’ paragraph, the machine vision system has these capabilities. The detailed-level matching considering the field of view and resolution requirements is carried out as was done earlier. Camera unit 1 is able to fulfil the given requirements.

However, the required accuracy for the inserting operation has also been defined. The accuracy of this operation is determined firstly, by the accuracy of the robot, and secondly, by the accuracy of the machine-vision hole positioning. The accuracy of the machine vision depends not only on the resolution of the camera, but also on the accuracy of the machine vision software, which again is dependent on the type of algorithm. Roughly, it can be said that, for example, a circle-fitting algorithm has 1/3 pixel accuracy, whereas a pattern-matching algorithm has 1 pixel accuracy. Once the detector resolution and the field of view is known, the spatial resolution of the camera unit indicating the distance on the object surface per one pixel can be calculated. In this case, the spatial resolution of the camera is 0,127 mm/pix and the pattern-matching algorithm has 1 pixel accuracy. However, the rule of thumb says that the recommended resolution should be 4-10 times better than the camera accuracy. Therefore, the positioning accuracy of the camera system can be estimated as 0,5 mm. It has to be noted that this is just a rough estimate and is affected by a number of factors which need to be measured in order to get more realistic values. Therefore, the accuracy is already given in the resource capability description and not reasoned according to the rules.

RULE 4a: Is the accuracy of the inserting capability (robot + machine vision system) enough?

```
IF process.getParam(“required_accuracy”) >= (movingWorkspace.getParam(“accuracy”) +
   positionRecognition.getParam(“accuracy”))
THEN
   RETURN TRUE
```

The rule shows that the combined accuracy of the machine vision system and the screwdriver robot is, in the worst case scenario, 0,8 mm. This was the original requirement, which means that the accuracy requirements are fulfilled.

5) Fasten the screw

The high-level capability mapping finds one device combination, namely the screwing robot consisting of the screwdriver unit and the Cartesian manipulator, which has the capability "screwing". The detailed matching based on the capability matching rules needs to check whether the screw type and size, as well as the required torque, are suitable for the existing screwdriver.

RULE 5a: Can the screws be screwed with the available screw driver?

```
IF screw.getParam("type") =  screwingHead.getParam("type") AND
   screw.getParam("size") <= screwingHead.getParam("screw_size_max") AND
   screw.getParam("size") >= screwingHead.getParam("screw_size_min") AND
   screw.getParam("torque") <= spinningTool.getParam("max_torque") AND
   screw.getParam("torque") >= spinningTool.getParam("max_torque")
THEN
   RETURN TRUE
```

The matching shows that the screws can be fastened with the existing screw driver. However, the screw driver is only suitable for one size of screws. If the screw size changes, it will immediately require physical adaptation to the system. This will be discussed in the following chapter.

The capability matching actions discussed in the previous examples can be done automatically, based on the rules in the rule-base. However, the results need to be validated by the human designer, because multiple simplifications have been made in the rules. In particular, the machine vision system has multiple parameters which affect the result of the positioning accuracy. It is impossible to take all of these into consideration in the automatic matching, and therefore human expertize really is required. Thus, in this thesis, only the most relevant parameters of the vision systems are considered in the rule-base.

5.3.4. Adaptation scenarios with the TUT-microfactory

As explained in Chapter 5.3.3, the current TUT-microfactory for flexible screwing was capable of fulfilling the requirements set by the cell phone screwing requirement. In other words, the system was fully compatible with the requirements of the product. This chapter will analyse what kind of adaptation actions are required to the existing system in the case of different change stimuli, and how the compatibility, adaptation effort and utilization evolve with the changing requirements. The following paragraphs discuss a few scenarios in which the requirements targeted at the production system change. The original cell phone scenario is referred to as "product A", and the subsequent scenarios are named "product B", "product C" and so on. Three different change stimuli will be considered: change in the product design; change of product; and change in the order requirement. In addition to those changes coming from the product and order side, eventually changes in the behaviour of the system itself will also be considered.

Change in the product design

This section considers changes in the product design. Three different scenarios are viewed: change in the hole locations – Product B; change in the screw size – Product C; and change in the accuracy requirement – Product D. The following table, Table 32, presents first of all the components of the current system and its capabilities. Secondly, it displays the capability requirements of the different product scenarios, from A to D, at the capability concept name level. The numbering behind the required capabilities indicates the existing capability that fulfils the requirement. Required capabilities which don't exist in the current system or which need some change are written in *italics and underlined*. Figure 67 will then display the capability requirements of the product scenarios B, C and D compared to product A in the form of Venn diagrams. The figure shows the common requirements zone, which represent similar capability requirements in the two products under comparison. The size of this zone is determined by the factors discussed earlier in Chapter 4.7.1.

Table 32. Existing system capabilities and capability requirements of products A, B, C and D.

System	Capabilities	Product A requirements	Product B requirements	Product C requirements	Product D requirements
TUT-microfactory frame Screwing robot Feeding system Belt conveyor Machine vision system 1 Machine vision system 2	attachmentFrame (1) screwing (2), pickingUp (3), transporting (4), inserting (5) plateFeeding (6) transporting (7) objectRecognition (8), positioning (9) objectRecognition (10), positioning (11), orienting (12)	Module frame (1) Transporting (7) Bulk feeding (6) PickingUp (3) (ObjectRecognition (10), Orienting (12), Positioning(11)) Inserting (5) (Object recognition (8), Positioning (9), Transporting(4)) Screwing (2)	Module frame (1) Transporting (7) Bulk feeding (6) PickingUp (3) (ObjectRecognition (10), Orienting (12), Positioning(11)) Inserting (5) (Object recognition (8), Positioning (9), Transporting(4)) Screwing (2)	Module frame (1) Transporting (7) Bulk feeding (6) PickingUp (3) (ObjectRecognition (10), Orienting (12), Positioning(11)) Inserting (5) (Object recognition (8), Positioning (9), Transporting(4)) *Screwing (physicModif 2)*	Module frame (1) Transporting (7) Bulk feeding (6) PickingUp (3) (ObjectRecognition (10), Orienting (12), Positioning(11)) Inserting (5) (*Object recognition*, Positioning (9), Transporting (4)) Screwing (2)

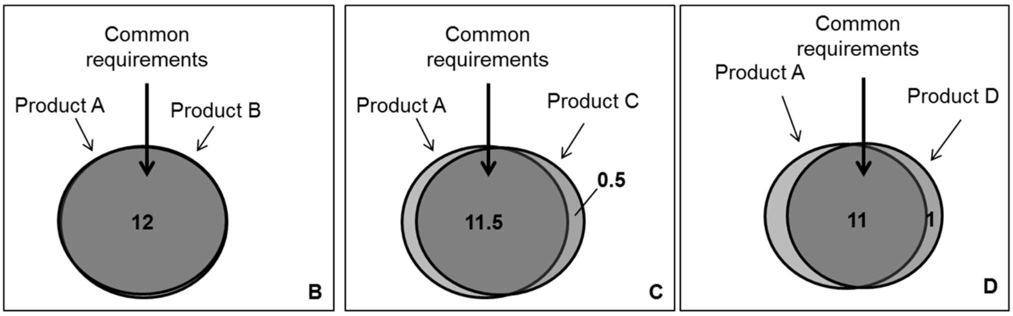


Figure 67. Common requirement zone of the product scenarios B, C and D with product A.

Change in the hole locations – Product B

In this scenario, the distance between the screw holes changes from 52mm to 70mm. Change in the hole locations doesn't cause any changes to the required capabilities at the concept name level, but the workspace size requirements for the "transporting" and "objectRecognition" capabilities change. However, if the product size remains the same, the change in the hole locations doesn't affect the system. This is because the original requirements for the workspace were defined based on the dimensions of the phone and the conveyor, as the position and orientation of the product on the conveyor were not fixed. Therefore, it can be said that even if the product design changes slightly, in practice the requirements targeted at the systems don't change in this case. The robot workspace covers the whole conveyor area, and the machine vision workspace was defined based on the outer dimensions of the cell phone. Both workspaces are therefore sufficient to cover any changes in the hole locations. The system is fully compatible with the new requirements.

Compatibility, effort and utilization of the existing system for the product B

ReqCaps = 12; ExistCaps = 12; AllSystemCaps = 12
→ Compatibility = 100 %; Effort = 0; Utilization = 100 %

Change in the screw size – Product C

In this case scenario, the screw size changes from M1.6 to M2. When changing the screw size, the capabilities remain the same at the concept name level. Only the screw size parameter is changed. The current screwdriver is suitable for screwing size M1.6 screws, but only that size screw. If the screw size changes, a new screwing head need to be attached to the screwdriver. The following rule is used to define whether the new screwing head is suitable for the given screwdriver.

RULE 6a: Attaching screw driver + screwing head

```
screwdriver = resource.hasCapability(“spinningTool”)
screwinghead = resource.hasCapability(“screwingHead”)

IF (spinningTool.getParam(“diameter_max”) >=
    screwingHead.basicDeviceInfo.getParam(“diameter”))
THEN
  RETURN TRUE
```

If the screwdriver head is not compatible with the screwdriver, for example in this case if the screwing head is for bigger screws than M2, the whole screwdriver needs to be changed. Small changes in the product design can in this case be handled with relatively small changes in the system. However, this example well illustrates that when certain border constraints of the capabilities are crossed, the magnitude of the adaptation can grow significantly due to the propagation of the changes. In this example case, as long as the required screw size is between M1 and M2, the system can be adapted by just changing the screwdriver head. If bigger screws are used, the whole screwdriver needs to be replaced. And again, these changes may be propagated to the robot if, for example, the new screwdriver is not compatible with the robot interface.

Compatibility of the existing system with the product C

ReqCaps = 12; ExistCaps = 11; PhysModifCaps = 1; AllSystemCaps = 12

→ Compatibility = 95.8 %; Effort = 4.2 %; Utilization = 95.8 %

Change in the accuracy requirement – Product D

For the original product, product A, the required accuracy of the screw insertion was set quite low. It was 0.8 mm, which meant that some amount of misplacements occurred and that those were tolerated. The current system barely fulfilled this accuracy requirement. In this product scenario, no misplacements are tolerated, and therefore the accuracy of the inserting operation needs to be increased to 0.4 mm.

The change in the accuracy requirement doesn’t affect the required capabilities at the concept name level, but the accuracy parameter of the inserting capability is changed. Based on the rules used in Chapter 5.3.3, the accuracy of the insertion depends on both the accuracy of the robot and the accuracy of the machine vision system. The accuracy of the insertion can then be increased by replacing the robot with a more accurate one, replacing the camera with one with better resolution or using a different type of machine vision algorithm which is more accurate. It is also possible to bring the camera closer to the measured object, in order to increase the pixels/millimetre ratio, or to change the optics with a greater focal length, again to decrease the field of view to increase the pixels/millimetre ratio. However, if the product size remains the same, it is not safe to decrease the field of view and therefore the two last options are not useable. So, basically, the options are to either change the robot, the camera, or the machine vision algorithm. In this case, the fastest and cheapest option is probably to change the camera, because cameras with better resolution are easily available at reasonable prices. Merely changing the algorithm probably wouldn't increase the accuracy enough.

Compatibility of the existing system with the product D
ReqCaps = 12; ExistCaps = 11; AllSystemCaps = 12
→ Compatibility = 91.7 %; Effort = 8.3 %; Utilization = 91.7 %

Change of product

In this category, three different product change scenarios are considered: new different parts added to the product – Product E; screws replaced with new parts – Product F; a completely different product – Product G. Their capability requirements are shown in Table 33 and the differences between their capability requirements and those of product A are shown in the Venn diagrams in Figure 68. It is known that a change in the product to be manufactured will most probably lead to a requirement for new capabilities for the system. This means that physical adaptation is required. In this presented case study, the user has given high priority to the re-use of old equipment. This means that any changes to the system should be implemented so that as much as possible of the existing system is retained.

Table 33. Existing system capabilities and capability requirements of products E, F and G.

System	Capabilities	Product E requirements	Product F requirements	Product G requirements
TUT-microfactory frame	attachmentFrame (1)	Module Frame (1)	Module Frame (1)	*Heating material*
		Transporting (7)	Transporting (7)	*Injecting material*
Screwing robot	screwing (2), pickingUp (3), transporting (4), inserting (5)	Bulk feeding (6)	*Feeding (physicModfi 6)*	*Molding*
		PickingUp (3) (ObjectRecognition (10), Orienting (12), Positioning(11))	*Disensing glue* (Transporting (4), ObjectRecognition (8), Positioning (9))	*Cooling*
Feeding system	plateFeeding (6)			*Opening the mould*
Belt conveyor	transporting (7)		*PickingUp*	*Releasing the part*
Machine vision system 1	objectRecognition (8), positioning (9)	Inserting (5) (Object recognition (8), Positioning (9), Transporting (4))	Inserting (5)	*Transporting*
Machine vision system 2	objectRecognition (10), positioning (11), orienting (12)	Screwing (2)		
		Module frame		
		Transporting		
		Disensing glue (Transporting, ObjectRecognition, Positioning)		
		Feeding		
		Picking up		
		Inserting		

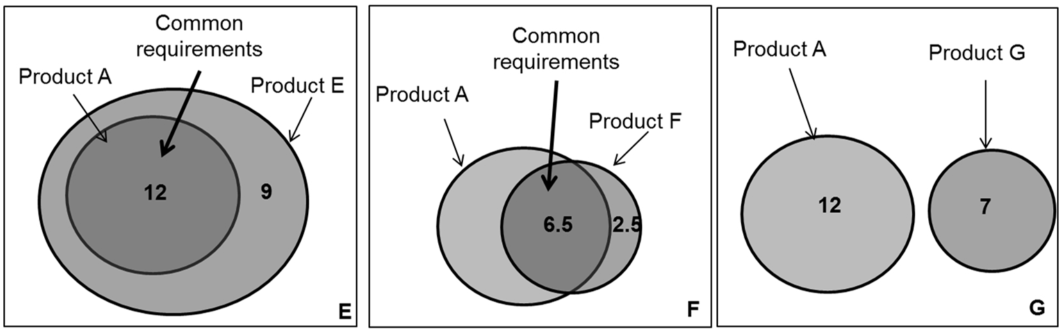


Figure 68. Common requirement zones of product scenarios E, F and G with product A.

New different parts added to the product – Product E

Product E is similar to product A, except that at the end of the process two of the screws need to be covered with small plastic parts, which are glued over the screws. This means that the old capabilities of the system are still needed, but new "gluing" and "feeding" capabilities need to be added. Furthermore, the current system's grasping capability doesn't match the requirement for grasping the plastic part, and therefore additional "picking" up and "inserting" capabilities are needed. Because of the small size of the TUT-microfactory module, not all of this can be included in the same module. This means that the existing system can still be used as a whole, but a new module with the required equipment should be added. Altogether, the required extra capabilities are: transporting the product to another module; detecting the screws and their position in the phone; dispensing glue onto the screw; feeding the plastic parts; picking up the plastic part, and inserting the plastic part into the screw hole.

Compatibility of the existing system with the product E
ReqCaps = 21; ExistCaps = 12; AllSystemCaps = 12
→ Compatibility = 57.1 %; Effort = 75 %; Utilization = 100 %

Here, it is assumed that the feeding of the plastic parts is handled with dedicated trays and no new machine vision system is required to recognize the parts and their position and orientation.

Screws replaced with new parts – Product F

Product F is assembled with snaps, so screws are no longer needed. Instead, the plastic parts need to be glued and inserted into the screw holes to cover them. Therefore, the "screwing" capability is no longer needed but a new capability, "gluing" needs to be added to the system. In addition, the feeding of the plastic parts needs to be handled. As the user-defined criterion was that the existing system should be used as much as possible, the screwdriver unit will be replaced with a dispenser unit and attached to the Cartesian robot. The robot also needs to be equipped with a gripper for grasping the plastic parts. For feeding the parts, the same tray feeder can be used merely by changing the tray or plate suitable for the plastic parts.

Compatibility of the existing system with the product F
ReqCaps = 9; ExistCaps = 6; PhysModifCaps = 1; AllSystemCaps = 12
→ Compatibility = 72.2 %; Effort = 20.8 %; Utilization = 54.2 %

Completely different product – Product G

In this scenario, the requirement is to manufacture the cell phone covers. From the system's point of view, this is a completely different product. It requires completely new capabilities, such as injection molding, and none of the existing capabilities are useable for this new requirement. Therefore, it can be said that the current system has zero compatibility with the new requirement.

Compatibility of the existing system with the product G

ReqCaps = 7; ExistCaps = 0; AllSystemCaps = 12

→ Compatibility = 0 %; Effort = 58.3 %; Utilization = 0

Change in the order requirements

In the order requirement category, two different scenarios are considered. These are: fewer capability requirements – Product H; and change in the production volume – Product I. Their capability requirements are shown in Table 34 and the differences with the capability requirements of product A are shown in Figure 69.

Table 34. Existing system capabilities and capability requirements of products H and I.

System	Capabilities	Product H requirements	Product I requirements
TUT-microfactory frame	attachmentFrame (1)	Module frame (1)	Module frame (1)
Screwing robot	screwing (2), pickingUp (3), transporting (4), inserting (5)	Transporting (7) ObjectRecognition (8)	*Transporting (paramModif 7)* Bulk feeding (6)
Feeding system	plateFeeding (6)		PickingUp (3) (*ObjectRecognition,* Orienting (12), Positioning(11))
Belt conveyor	transporting (7)		
Machine vision system 1	objectRecognition (8), positioning (9)		Inserting (5) (Object recognition (8), Positioning (9), *Transporting (paramModif 4)*)
Machine vision system 2	objectRecognition (10), positioning (11), orienting (12)		Screwing (2)

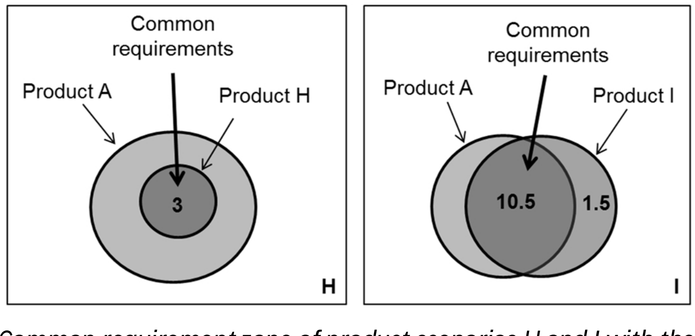


Figure 69. Common requirement zone of product scenarios H and I with the product A.

Fewer capability requirements – Product H

In this scenario, the screws have already been screwed to the cell phone in the previous process phase. Now the order requirement is only to check that all four screws are in their correct place. No new capabilities are needed, and most of the existing capabilities are no longer used.

Compatibility of the existing system with the product H

ReqCaps = 3; ExistCaps = 3; AllSystemCaps = 12

→ Compatibility = 100 %; Effort = 0 %; Utilization 25 %

Increase in production volume – Product I

In this scenario the production volume grows so that 30% more products should be produced in the same time that was originally used for producing product A. This means that a new cycle time for the product needs to be defined and the current system's ability to cope with this cycle time requirement needs to be evaluated. In case A, the system was not running at its full speed. Both the robot and conveyor speeds could be increased by half. However, how fast the robot can move with the screwdriver can only be discovered by testing. Due to the weak accuracy of the inserting operation, the misplacement of the screws consumed a lot of extra time to accomplish the process. Therefore, by increasing the accuracy of the insertion, the production volume can be increased. Three adaptation actions, one physical and two parametric, are needed to achieve the required new production volume. These are: changing the camera, and adapting the speed of the conveyor and the robot.

Compatibility of the existing system with the product I

ReqCaps = 12; ExistCaps = 9; ParamModifCaps = 2; AllSystemCaps = 12

→ Compatibility = 87.5 %; Effort = 12.5 %; Utilization = 87.5 %

Change in the behaviour of some system component

As discussed earlier in this thesis, the capabilities (behaviour) of the devices may change during their individual lifecycles. In this case scenario it is noted that the accuracy of the robot has been decreased from 0.3mm to 0.6mm. This new updated capability information is assigned to the robot. Now it is noted that the capability no longer satisfies the given accuracy requirement for the inserting capability. This means that either the robot or the camera needs to be replaced to increase the accuracy. However, such a drastic change in the robot's accuracy indicates that bigger problems may be coming. Therefore, it may be wiser to replace the robot with a new one. This example illustrates that changes in the resource capability may affect the compatibility of the resource to the given requirement and therefore require adaptation actions; in this case changing the robot.

Compatibility of the changed system (system z) with the original product A

ReqCaps = 12; ExistCaps = 11; AllSystemCaps = 12

→ Compatibility = 91.7 %; Effort = 8.3 %; Utilization = 91.7 %

In this case the effort indicates the effort of changing the system back to be again compatible with the requirements of product A.

Compatibility domain graphs of the product scenarios

Figure 70 shows the "compatibility – requirement change" graph of the previous scenarios defined according to the formulas (2) and (5) presented in Chapter 4.7.1. In practice, the compatibility shows what percentage of the product requirements can be satisfied with the existing system without any adaptation actions for each product scenario. In other words, it illustrates the relative size of the intersection between the product requirements and the system capabilities where the operation of the system can continue without modifications. Very different products, such as product G, are not really comparable with the others. It was included in the figure to show what happens if all the requirements targeted to the system change. In such cases, the compatibility falls to zero. Product H had fewer requirements than the original product, and no new capability requirements. This is a special case and is therefore drawn on the negative side of the x-axis. Even though, within this case set, the result is a descending graph as the magnitude of the change in requirements grows, this is not always the case, as discussed in Chapter 4.7.1.

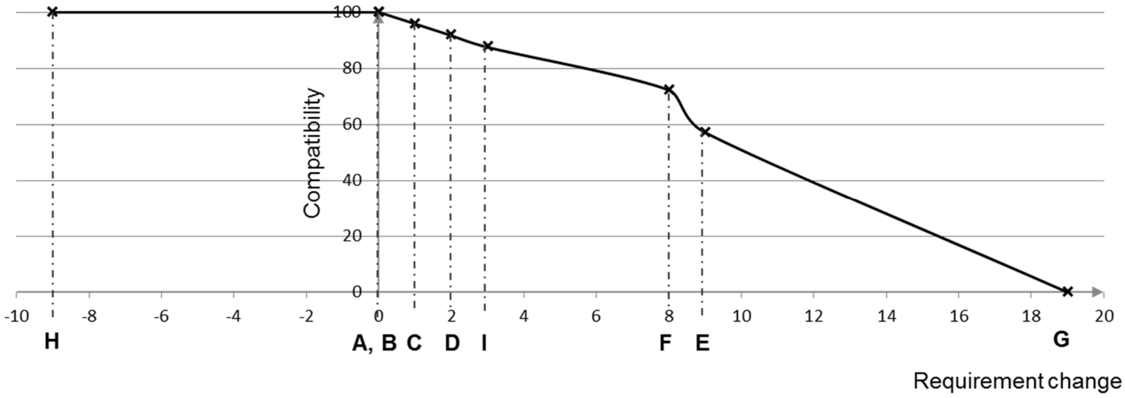


Figure 70. Change in the compatibility within the different product scenarios.

In order to compare the relative effort of the needed adaptation between different product scenarios, the "compatibility – effort" graph was drawn, as shown in Figure 71. This graph clearly shows that the compatibility itself doesn't reveal much about the effort needed to modify the system when comparing very different products, i.e. products having different amounts of capability requirements. Examples of this are the products F, E and G, whose amounts of required capabilities differ notably.

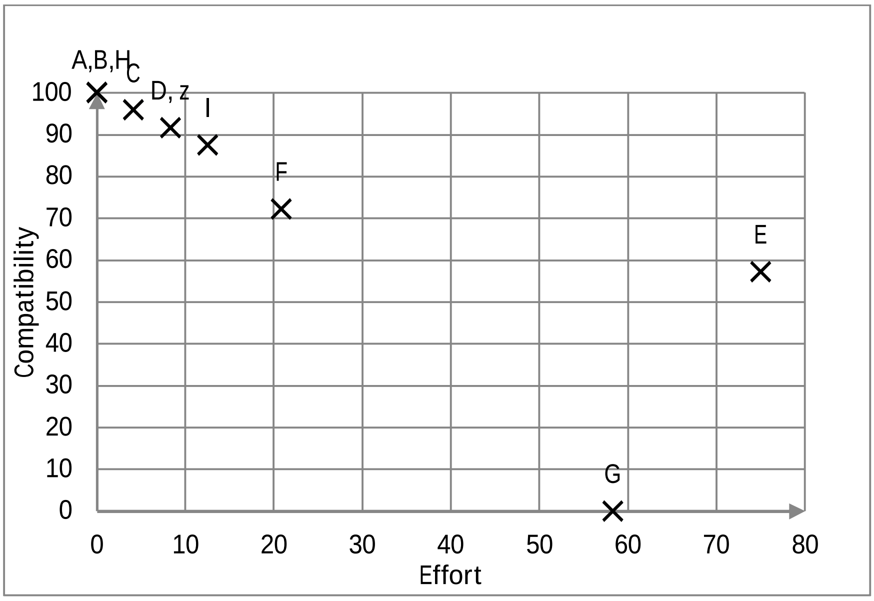


Figure 71. Compatibility versus effort.

Finally, the "compatibility – utilization" graph was drawn to be able to compare the product scenarios based on their compatibility and the re-usability of the existing system. The criteria for the analysis was discussed in Chapter 4.7.1. The scenarios falling into the upper right corner represent "good cases", for which the compatibility is good and most of the system capabilities can be utilized. On the other hand, the scenarios in the lower left hand corner represent the "worst cases", for which no capability requirements can be satisfied with the current system and therefore the current system cannot be utilized. Scenario H represents a case where all the required capabilities exist in the current system, but only small number of all the capabilities offered by the system are used. The rest of the capabilities are "wasted". In contrast, scenario E represents a case where all the offered capabilities can be utilized, but more capabilities need to be acquired in order to satisfy all the requirements.

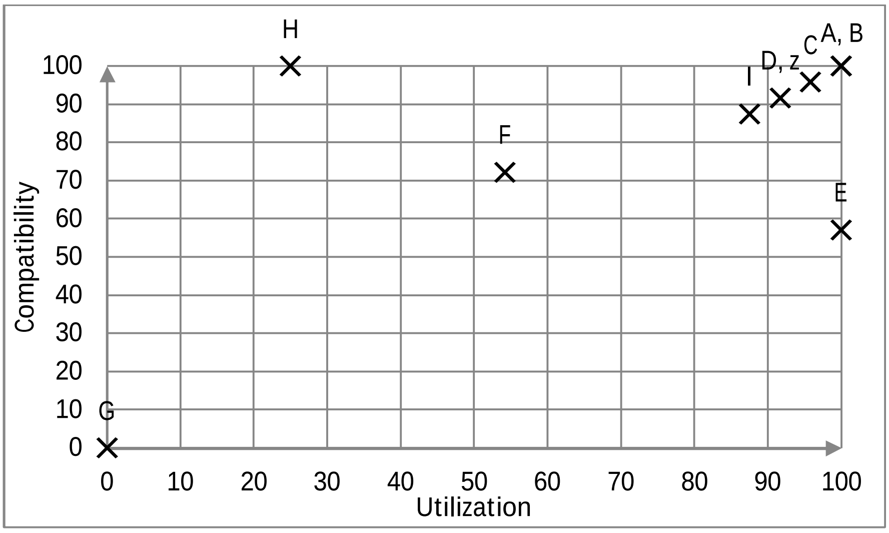


Figure 72. Compatibility versus utilization.

As was discussed in Chapter 4.7.2, the compatibility domain approach is highly subjective and dependent on the context. The input information used to draw the graphs, e.g. the number of required, provided and matching capabilities, depends on the level of granularity used for the capabilities. The evaluation of the propagation of the changes in the system, and again the evaluation of the needed new capabilities, is based on human reasoning, and is therefore highly subjective. Thus, even though the graphs give quantitative estimates of the compatibility, effort and utilization, the estimates should be considered more as qualitative than quantitative analyses. They don't provide absolute values for the impact of the changes, but they do enable relative comparisons between similar cases.

6. DISCUSSION

6.1. Evaluation of the proposed methodology

The objective of this thesis, defined in Chapter 2.2 was: “to develop a methodology to support computer-aided adaptation of production systems in a changing environment”. This high-level objective was divided into four sub-objectives: 1) definition of an adaptation schema; 2) development of a resource description model based on capabilities; 3) development of framework and rules for matching the product requirements against resource capabilities; 4) creation of a preliminary approach to evaluating the impact of changes in product requirements on the production system. This chapter is devoted to describing and analysing how well the objectives set for this thesis were achieved. It can be stated at the outset that, on the whole, each of these objectives was satisfied. In the following sections, the solutions for these objectives are analysed one by one, and the recognized limitations are discussed. After that, there are some general remarks about the developed methodology as a whole.

Sub-objective 1: Definition of the adaptation schema

The adaptation schema was developed and modelled in the form of activity diagrams using the IDEF0-modelling language. The schema provides a comprehensive explanation of the adaptation process by clarifying the activities, information flows and required resources during the adaptation planning and reactive adaptation process. The schema provides the backbone for the whole adaptation methodology and guides the use of the other elements, which have been developed, during the adaptation planning and reactive adaptation.

The adaptation schema was intended to support both human-centric adaptation planning and dynamic reactive adaptation. This is satisfied, because the schema does not constrain “who” or “what” should perform the reasoning steps. Neither does it compel the reasoner to go through all the adaptation steps or use all the indicated resources. For example, in the case of dynamic adaptation in the holonic system demonstration, no physical adaptation is required and therefore some activities from the schema can be dismissed. On the other hand, all the information resources indicated in the schema may not be always available. The accuracy of the result is therefore highly affected by the availability and quality of the input information. Particularly in the case of automatic reasoning, the unavailable or faulty input information can produce misleading results. Therefore, human involvement in the final decision making is assumed. This will be discussed in more detail later, in the general remarks.

Even though the schema contains more inputs, information resources and controls than have actually been applied in the case examples, the level of detail of the activities is the same as in the implementation. This means that the resource description and capability matching framework are designed so that they support all the activity steps included in the schema. As discussed earlier in Chapter 4.6.2, each of the activities could be further decomposed into lower, more detailed-level activities. Currently, the adaptation schema doesn’t go into the level of detail whereby the systems could be evaluated in terms of their ability to answer, for example, the volume and cycle-time requirements or to calculate the cost of acquiring and operating those resources. Some of the information needed for that kind of reasoning is included in the resource description which enables a human expert to assess these aspects. However, new algorithms and rules should be developed in order to automate such reasoning.

Sub-objective 2: Development of a resource description model based on capabilities

The overall resource description and capability model were developed in order to describe the resource capabilities and characteristics so that they facilitate the selection of suitable resources for a given product requirement. The ontologies were utilized to provide formal computer-interpretable models and to enable the computerized filtering, handling and management of the resource information. The objective with the resource model was to be able to describe the combined capabilities of multiple co-operating resources. The capability model which was developed allows this to happen at the capability concept name level, whereas the combined capability rules enable this at the parameter level.

The resource description was intended to be used in the context of the adaptation process. Therefore, it was important to incorporate the lifecycle aspect into the resource description. As discussed earlier, the production environment is constantly evolving and the behaviour and properties of the components constituting the overall system change during their individual lifecycles. Without up-to-date information about the system, it is difficult to make decisions regarding its adaptation, or at the very least, the decisions won't be based on trustworthy evidence. In order to provide more reliable resource information, the proposed resource description approach has two representations of the resource, one presenting its nominal capability and other presenting its actual, updated capability.

At present, the capability model covers the capability instances needed in the case studies (and similar environments), i.e. capabilities existing in the TUT-machining laboratory and the TUT-microfactory environment. The developed capability model is freely extendable and new capabilities can be added to the model when needed. It is general enough to be applied to various different domains. Even though it wasn't within the scope of this thesis, the model could also be used for modelling human competences.

The developed capability description approach differs from existing semantic resource description approaches from its very inception. From the outset, it was defined so that it would support combined capability modelling and the re-usability of the capabilities for different kinds of resources. This has strongly affected the structure of the capability model, especially in terms of the allocation of the parameters to the specific individual capabilities. The properties of the resources are distributed under a number of separate simple capabilities, which may initially cause a little confusion when describing the resources with this model. Such confusion can be illustrated with the following example for describing a milling machine's capabilities. The milling machine's properties given in the suppliers' catalogue cannot be assigned directly to the machine, but need to be divided into multiple separate simple capabilities. As the milling machine's main function is to spin the tool in order to cut material, its main capability is "spinningTool". However, not all the machine's properties can be assigned to this main capability, because, for example, a screwdriver also has that "spinningTool" capability, yet it definitely doesn't have all the same properties as a milling machine. Therefore, other capabilities also have to be assigned to the milling machine, such as "movingWorkspace" and "degreesOfFreedom". This may not be the most intuitive approach when assigning the capabilities. However, if all the properties were to be assigned in one group for one capability, it would resemble traditional resource description approaches, and would lose its expressiveness and its ability to model combined capabilities.

So, the resource description approach proposed here differs from traditional resource description approaches, as they don't usually have enough expressiveness to describe multifunctional resources. While the traditional approaches tend to classify the resources into groups based on their properties and functionality (e.g. milling machine, lathe, conveyor, etc.), the approach used here classifies the capabilities and assigns those to the resources. In this way, one resource can have multiple capabilities to be used in different contexts, and new capabilities can be assigned to the resource as they emerge. Another advantage of describing the resources based on their capabilities, rather than based on their type, is that one doesn't need to assume, for example, that a milling machine has all the capabilities of a normal milling machine, but merely to assign the capabilities relevant to that specific resource. This modular capability-modelling approach therefore provides more expressiveness and flexibility for describing different kinds of resources. The modular approach for assigning the capabilities to the resources also provides significant support for adaptation planning. Describing each resource in the system based on its simple capabilities, rather than describing the complete system as a whole, makes it easier to change and replace components in the system.

One final comment, which actually relates to both sub-objectives 2 and 3, needs to be stated. The technical accuracy of the capability descriptions developed here, and of the rules, could probably be improved through greater expertise and experience with the technology and processes used as examples. A wide range of different processes were covered, although at times the author lacked any in-depth technical knowledge of them. Despite consultation with experts about the processes, it must be acknowledged that there may still be room for improvement, especially with regard to assigning the parameters to the simple capabilities and in developing the capability matching rules. However, as the original goal was not to develop a perfect resource description and rules, but rather to develop a concept and a framework for describing and matching the capabilities to support adaptation, this deficiency may be regarded as acceptable.

Sub-objective 3: Development of a framework and rules for matching the product requirements against resource capabilities

A conceptual framework for matching the product requirements against the system capabilities was successfully created. Its main elements are the capability taxonomy, which enables mapping between the product requirements and the system's capabilities at the capability concept name level, and the capability matching rules, which facilitate the detailed matching. As with the capability model, the rules were designed to serve two case environments, the TUT-machining laboratory and the TUT-microfactory. If applied to a different environment, new rules would need to be implemented.

The main goal of the rules was to show that the information saved with the formal resource description and capability model can be used for automatized capability-matching and for generating different configuration scenarios. The capability matching rules prove that such a resource description can be used to match the product requirements with suitable resources. The fact that it was possible to create rules that compare the product and resource information saved in the ontology, and provide feasible matches, proves that the resource description and capability model provide enough expressiveness for such use cases. Therefore, the rules which were developed for these cases validate the developed resource description and capability model, and just as importantly, prove that automatic matching is possible.

At this point in time, only part of the rule-base is implemented as part of the modular ICT-environment. This is mainly due to a shortage of programming resources. However, the rule-base is under way, and new funding has been applied for in order to get it properly implemented and integrated into the fully working, modular ICT-environment. Because of the incomplete implementation of the rule-base, it is not yet possible to test how well the capability matching works in practice in more complicated cases; for example, how long would the reasoning procedures take. Nevertheless, as the actual implementation of the rule-base was not defined as the goal of this thesis, the fact that it is at present incomplete does not reflect on the validity of the thesis.

Although the developed rule-base does enable automatic reasoning, human intervention is still required. It should be remembered that the goal of the developed adaptation methodology was not to make everything automatic, but to keep humans involved in the decision making and control loop. In practise, this means validation of the automatically generated scenarios and selection of the most desirable solution based on the user-specific criteria. The capability model and combined capability rules which are proposed here cannot provide perfect descriptions of the real capabilities of the system. It is recognized that the definition of the combined capabilities in detail at the parameter level is very difficult with the digital models. For example, the detailed definition of a workspace of a combination of multiple resources, even the relatively simple robot + gripper combination, requires complicated spatial reasoning with mathematical models, and this was beyond the scope of this thesis. This problem is due to the fact that many of the properties emerge as a behaviour of the system as a whole in a specific context, and they cannot be decomposed to the properties of the various components. For instance, in the case of a machining operation, the surface finish may depend on chattering which in turn depends on multiple factors, such as the structure of the machine, the control of the machine, the characteristics of the tool, the parameters of the specific operation, the characteristics of the material, and so on. Therefore the only way to predict chattering would be to have a comprehensive model of the machine, its control, and the manufacturing process, since the behaviour cannot be traced back to its components. This was beyond the scope of this thesis. Another example of a parameter, which is difficult to define in detail, is the accuracy of a machine tool, as discussed already in Chapter 4.4.4.

Therefore, from a logical point of view, the proposed approach can be used for determining those properties of combined capabilities which can be traced back to the simple capabilities, i.e. to the individual components of the system. It is not suitable for determining those properties of combined capabilities which cannot be decomposed to the properties of the various components, as exemplified above. It is precisely because this problem was already recognized in the early phases of the research that so much emphasis was placed on involving human expertise in the final decision making. Therefore, certain simplifications in the combined capability rules were also permissible. However, the rule-base can be extended incrementally, which will eventually lead to ever more accurate and realistic reasoning results, which will require less human intervention.

When matching the resource capabilities with the product and order requirements, there may not always be a one-to-one match. For example, a specified manufacturing volume may be achieved by using only one fast machine or two slower machines. If a certain accuracy is required, a feasible result may be obtained by using inaccurate devices and improved quality detection at the end of the process. As it stands, the automatic capability matching is not able to take this kind of fact into account. To do so would mean that the compatibility of the devices would have to be evaluated using some kind of fuzzy logic rules. These would allow those kinds of resources whose properties don’t all have a 100% match with the requirements to be added to the proposal list. And of course, a human expert could also make these fuzzy compatibility evaluations. The current approach doesn't allow this, because the capability matching dismisses those capabilities which don't completely fulfil the requirements.

Finally, there is one limitation in the current capability matching framework which needs to be recognized. The capability matching and the capability model do not take into consideration the effect of the human worker operating the resource. For example, a well trained and experienced operator may be able to manufacture the same product much faster and with better quality than an inexperienced operator working with a similar machine. The accuracy of machining operations, in particular, can easily be affected by the professional skills of the operator. As the human aspect can significantly affect the capability performance that can be achieved, this aspect should be integrated into the methodology in the future. As with the previous point, this aspect would also require some sort of fuzzy reasoning.

Sub-objective 4: Creation of a preliminary approach to evaluating the impact of changes in product requirements on the production system

A compatibility domain approach was created, in order to estimate the impact of product requirement changes on the existing production system. The compatibility domain approach examines the topic from three aspects: the compatibility of the existing system to the new requirements; the relative effort entailed in making the needed modifications to the system; and, the utilization (re-usability) of the existing system for the new requirements. The compatibility domain approach utilizes the resource capability descriptions and the capability matching framework, in order to identify whether, and if so how many, new capabilities need to be added to the existing system when the product requirements change. The compatibility domain approach offers a valuable aid to decision making, when comparing different product scenarios which require change. It rationalizes the decision making, because it allows easier identification and allocation of the needed changes to the system, and therefore also opens up the possibility for more realistic evaluation of the costs of adaptation.

The restrictions of the compatibility domain approach were already discussed in Chapter 4.7.2 and are not repeated in detail here. Nevertheless, it must be acknowledged that the approach is just a preliminary, highly simplified concept and has multiple restrictions. The validity of the comparison of different product scenarios depends strongly on the similarity of the compared products. In order to evaluate the real effort and cost of the required adaptation, the nature of the specific adaptation activities needs to be considered. In its present state, the compatibility domain approach estimates the effort purely based on the number of new capabilities that need to be added to the system, and the system ramp-up, for example, is not considered. This is a significant omission, because ramp-up is one of the biggest costs when a change is introduced to a system. In complicated cases, the system ramp-up may require months to be completed, and during the ramp-up phase either the production rate or the production quality (or both) may be much lower than the required targets. Therefore, the incorporation of this aspect into the methodology should definitely be considered in the future. Having said that, at this point these flaws in the method are acceptable, because the original goal was to think up a preliminary approach to how the developed resource description and capability-matching framework could be used to facilitate decision making in terms of the impact of change.

GENERAL REMARKS ON THE DEVELOPED METHODOLOGY

The developed methodology supporting computer-aided adaptation of production systems should not be directly compared to other available methodologies, because their viewpoint and scope are so different. A wide range of different methodologies supporting adaptation were presented in Chapter 3.3, and their limitations were discussed there. The starting point for this work was to try to overcome some of those limitations. As the goals were different, they cannot be directly compared with each other. One clear advantage of the resource description approach developed here, in comparison to the other available ones, is that it provides a solution for managing the combined capabilities of multiple co-operating resources. Previous, apparently similar, approaches haven't dealt with the parameters, but only with the skill (capability) names. Dealing with just the capability concept names doesn't provide enough information to select resources from multiple options, all having the same capability concept name. This limitation was overcome with this approach. Another advantage of this work is that it provides a holistic, integrated approach which supports adaptation from multiple viewpoints, all combined into one methodology.

The computer-aided adaptation methodology developed in this work is characterized by strong human involvement, especially in the final decision making. This is seen as important, particularly because of the volatile quality of the available input information. Even though humans are not able to make informed decisions without good input information, they still have more intelligence than a computer when it comes to spotting obvious mistakes that may exist in the input information, or that may occur during the reasoning procedures. The intelligence of the developed methodology lies in the fact that it utilises an efficient combination of human intelligence and computer processing power. The methodology aims to utilize computers for the tasks which they are good at, such as handling, processing and filtering large amounts of data. Humans, on the other hand, are responsible for making the intelligent decisions. The holonic demonstration presented a case where the "intelligent decision making" is done by the "holons", which can be either physical resources, software modules, or humans. The holons utilize their ability to negotiate with each other and to make decisions based on the acquired information. Algorithms (rules) are still needed by the holons to reason with that information. The intelligence of these holons (or the quality of the results they provide) can be increased by learning from experience. This learning is facilitated by collecting the usage history of the resources.

It has to be noted that the quality of the results produced by the developed adaptation methodology depends strongly on the available information resources. In order to make a match between product requirements and system capabilities, valid information about both of them has to be available for the reasoning. As in all reasoning, the quality of the result of the reasoning depends not only on the quality of the algorithm, but also depends strongly on the quality of the input information. If the description of the product requirements or system capabilities has errors or missing information, it will lead to a flawed reasoning result. As the creation of a resource description for an individual resource is a manual task, human errors may naturally occur. The descriptions of the product requirements were not in the scope of this thesis, but their role in the capability-matching must not be underestimated.

The adaptation approach taken in this thesis is reactive rather than proactive. The adaptation takes place after external changes are noted. Terkaj et al. (2009a) distinguished between planned changes and unplanned changes. Planned changes happen as a result of conscious managerial actions, whereas unplanned changes occur independently of the intentions of the company (Terkaj et al. 2009a). In the presented adaptation schema, the production environment is considered as dynamic and unpredictable, without any knowledge of the future. Only the unplanned changes which require adaptation are considered. However, if there is some known information about the future, then some degree of flexibility in the system could prove extremely beneficial. According to Terkaj et al. (2009a) flexibility tries to create a position in which the system is able to accommodate external changes without modifying its structure. In many cases, this is much more beneficial than adopting continuous reconfigurations. (Terkaj et al. 2009a.) In order to enhance the developed adaptation approach so that it is better suited for a real industrial context it should be extended so that it would include considerations of the future, if such information about the future, even uncertain, is available. The adaptation plans could then be created from a more strategic perspective, taking future requirements and possible scenarios into account, and thus trying to minimize the required changes to the system in the future. This means that the system could be adapted taking into account its flexibility for future situations, which are known or can be assumed.

The capability-based resource description and the framework for capability matching developed in this work support all the levels of adaptivity, namely physical, logical and parametric. Therefore, they can be used to support both reconfiguration and flexibility aspects. They allow the evaluation of the robustness of the existing system for possible future requirements, and identification of the adaptation needs for different product scenarios. Thus, if planned changes or external changes which can be forecast are taken into consideration during adaptation planning, it would require these aspects to be incorporated into the adaptation schema.

6.2. Contributions

This study resulted in a holistic, integrated adaptation methodology for production systems. The developed, capability-based adaptation methodology supports multiple aspects of production system adaptation, namely physical, logical and parametric adaptation in both static, human-controlled adaptation planning and dynamic, reactive adaptation contexts, as shown by the case studies. The methodology consists of the following novel elements:

1) Knowledge representation for describing production resources, their capabilities, and the combined capabilities of multiple co-operating resources;
2) A framework and rules for matching the resource capabilities with the capability requirements set by the product characteristics in the context of adaptation planning and the reactive adaptation process;
3) Adaptation schema defining the use of the resource description and the capability matching rules, as well as other information resources, as part of the overall adaptation process.
4) A preliminary approach to evaluating the impact of changes in the product requirements on the production system.

The core of the developed adaptation methodology, and therefore the main contribution of this thesis, is the formal resource description and capability model. These, together with the developed capability taxonomy and rule-base, support at least the following activities:

- Describing the capabilities of individual resources;
- Combining simple capabilities into higher-level, combined capabilities;
- Reasoning out the combined capability of multiple co-operating resources;
- Automatic matching of product requirements and resource capabilities;

- Finding resource combinations to fulfil a given product requirement;
- Checking whether the current system is able to fulfil a given requirement;
- Making sure that all the needed components exist in the system (combined capabilities);
- Showing the parameter range of the available capabilities, allowing decisions to be made about the parametric adaptation;
- Showing the available capabilities on the factory floor and their locations, thus providing the necessary information for the logical adaptation (e.g. re-routing).

As a result of the above, this approach allows the automatic generation of system configuration scenarios based on the given requirements. It also supports the rapid allocation of resources and the adaptation of systems. In addition to generating adaptation plans, the developed methodology allows the possibility of evaluating the robustness of the existing production system to possible changes. From the information management point of view, it supports automatic filtering of information and finding suitable solution proposals from a large search space, thus reducing the manual information-processing required during adaptation planning. In a small workshop with only a few resources, management of the resource information is not a problem. In large factories and production networks, automatic management and filtering of this information has substantial potential for reducing the amount of manual work, and thus reducing the time used for planning activities. In addition to providing a means for automatic reasoning, the developed resource description also provides additional information for human experts to carry out various planning activities relating to process planning and resource selection.

The capability description of the resources is a major element when searching for suitable candidate resources for different applications. Therefore, the developed resource description and the model for managing the combined capabilities of multiple co-operating resources support both the original system design and the adaptation phases. They are especially useful for adaptation, because they allow the adaptation planning to start by viewing the existing system as a whole entity, without having to consider each individual item of equipment comprising the system. The updateable resource description approach contributes towards better quality input information for making decisions during adaptation planning and reactive adaptation. The resource description and capability model also provide support for the original design of the system. During the original system design, they allow the search for those resources which together fulfil the capability requirements, thus obviating the need to break the capability requirements down into their most atomic elements.

Even though it was not one of the original goals of the thesis, the resource description model, the capability taxonomy and the capability matching rules can also be used to support DFMA (Design for Manufacturing and Assembly) and CE (Concurrent Engineering) methodologies. The main point in these methodologies is to consider simultaneously the product design, the process and the system where the product will be manufactured, and to try to design the product so that that it is easy to manufacture and assemble. With the help of the approach developed here, the product designer can take the existing system and available in-house devices into account when designing a product. For example, he/she may first draw the high-level process description and then search for the available technologies for, for example, fastening. The result may be that, for example, a riveting capability already exists on the factory floor. The designer then needs to evaluate whether riveting could be a suitable method for fastening the parts for that specific product.

Dynamic production environments put pressure on production systems to themselves become dynamic. Dynamic adaptation to changing requirements requires reactivity, which in turn calls for self-organising abilities from the system. In order for the system to be able to self-configure itself, the components comprising the system need to know and be able to describe to others what they are, and what they can do in the system. In other words, they need to be able to express their own capabilities. The developed resource description, capability model, capability taxonomy and capability matching framework and rules make a significant contribution towards such self-organizing systems. They give the entities an ability to describe themselves, advertise their capabilities, and to form combinations with other entities. They also enable the orders to express the capabilities they need, and finally to make a match between them. Therefore, assuming the entities are also able to negotiate with each other, they may autonomously organize the production based on the available capabilities.

Last but not least, the work developed in this thesis contributes significantly to evaluating the impact of changes in product requirements on a production system. This approach evaluated three aspects: 1) the current system's compatibility with different product requirements; 2) the relative adaptation effort (magnitude of change) needed to modify the system to be compatible with the requirements; 3) the utilization (re-usability) of the current system in the new product scenarios. These aspects were illustrated in the form of graphs, which allow the comparison of different product scenarios in terms of the impact that the required changes will have on the system. Even though the evaluation was rather simplified and crude, it is a first step towards quantifying the needed adaptation to the system. It allows rough comparisons to be made between different product scenarios requiring adaptation, and possibly the selection between different products and production strategies. The ability to evaluate how well the current production system fulfils the new product requirements, and the number and type of modifications that need to be made to the system, opens up the possibility of evaluating the investment needed to achieve the adaptation. This is very important when making decisions about whether the current system should be adapted, or a new one should be built from scratch.

6.3. Future work

This chapter discusses some future work and additional ideas and concepts which would further enhance the developed methodology. Many ideas for enhancement were already discussed in conjunction with the evaluation of the proposed methodology in Chapter 6.1 and are not repeated here in detail. These included, for example, consideration of the evolution of future requirements as part of the adaptation planning, consideration of the human factor for achievable resource capabilities, consideration of the ramp-up time and cost when evaluating the impact of change. The first and most urgent future task should be to finalize the implementation of the rule-base. Once that is done, the capability-matching can and will be tested with multiple different product scenarios. Based on the results of these tests, the resource description and rules will be further developed in order to increase the accuracy of the matching results. The following paragraphs introduce some other visions for future developments.

As discussed earlier, simulation can be used to validate the scenarios generated by the automatic reasoning methods. In particular, simulations should be used to analyse the combined workspace, reachability and possible collisions that might occur when multiple resources are cooperating. As creating simulation models manually is a time consuming task, automatic simulation generation would greatly enhance the developed methodology by significantly reducing the time needed to create the simulations, thus making them more feasible. The developed resource description supports the description of the position and orientation of the resources on the factory floor. Furthermore, the position and orientation of device interfaces are presented in the resource ontology as vectors. The simulation software could automatically attach the interface to the device model according to the saved vector information. Based on these interfaces, and the positional information about the devices on the factory floor, these could be loaded into the simulation world, allowing the system layout to be formed automatically. The matching of device interfaces could first be done based on the digital resource description. The simulation could then validate whether they really do match, e.g. that two conveyors are of the same height and able to co-operate. For example, the 3DCreate simulation software from Visual Components (Visual Components 2012) has an extensive library of robots and other system components from various manufacturers, in which the kinematics and functionality are pre-programmed. The resource description could be extended to include the Denavit-Hartenberg parameters (Wikipedia 2012) of the robots, and that information could be introduced into the simulation software, in order to facilitate a kinematic analysis of the system.

The methodology doesn't consider how the programs are generated for the machines. In order to route the product to a certain machine, it needs to have a program for performing the required activity. In the current implementation of the demonstration, the programs are pre-programmed for each machine. This is not, of course, a very adaptive approach. In the future, automatic generation of the tool paths and other machine programs could be integrated to form part of the dynamic operation environment framework. This can be done by integrating CAM (Computer Aided Manufacturing) software as part of the modular ICT-architecture. It would significantly enhance the proposed methodology and make it more useable in the dynamic adaptation context.

The creation of the resource description should be simplified. In its current form, the user needs to select suitable capabilities for the resources from the capability listing. Basically, he/she needs to know what kind of capabilities should be assigned to different types of resources. The Capability Editor could be enhanced so that it would suggest to the users the possible capabilities based on the capabilities of similar type of machines. This could mean the generation of some kind of resource description template, which would give hints of what kind of capabilities should probably be added.

The collection and use of the history data can increase the intelligence of the developed methodology by enabling learning from experience. However, the raw data is not useful without the context information, i.e. in what kind of applications the resource has been used. The condition of the resource largely depends on its usage history. In order to be able to estimate the condition and remaining lifetime of the resource, it is important to know the applications where it has been used and on what kind of process parameters it has been operating. The production system resources can take different roles based on their capabilities and the context in which they are used. For instance, a screwdriver can be used for screwing, while it takes the "screwdriver" role. It may also be used for drilling taking the "drill" role. In extreme cases it may even be used for hammering, when it takes the "hammering-tool" role. Naturally, the resource's performance will differ depending on the roles and contexts in which it is being used. The history data should be managed by the role engine, so that it can be associated with the specific roles that were used while collecting the data. In this way, the content and context information could be combined, enabling new knowledge to emerge. This knowledge can then be utilized towards a more successful adaptation. The information on how the resource behaved in a specific context, and in a specific role, could be later used for resource selection for similar applications. The role engine should be developed in order to manage this kind of history information.

7. CONCLUSIONS

The operational and business environment changes rapidly. Adaptation, both static and dynamic, is expected at all levels of operation, not only on single production facilities, but throughout distributed factories and logistic networks. As a result of this increasing process complexity, caused by the challenges discussed in Chapter 1.1, companies have to shift from traditional static optimization of production processes and systems to the adaptation of their systems and the dynamic distribution of the orders. Today's production systems can be regarded as having some characteristics of a Natural System. In such a system, the environment evolves over time and the system itself has to evolve during its lifecycle in order to survive in its environment. Optimization, in such a constantly changing environment, is no longer feasible, or even possible. It is of primary importance to find feasible solutions fast, in order to be able to adapt rapidly to the changing requirements. This new operating environment requires that real-time information about the production system and its components, the production process and the individual products is integrated into process, production and adaptation planning and control.

Based on the extensive literature survey, it can be stated that the ability to cope with a rapidly changing production environment and changing requirements can be enhanced in four complementary ways: 1) the development of adaptive hardware, for example modular system components with standardized plug-and-play interfaces, which allow the easy integration and exchange of system components; 2) the development of adaptive, self-configurable control systems; 3) the development of new methodologies and algorithms which guide and support the adaptation planning process at the whole system level; 4) the development of novel information models and information management systems to provide accurate, up-to-date information for adaptation purposes. The objective of this thesis was to provide solutions for the last two of these, (3 & 4).

The research carried out during this thesis produced a computer-aided, capability-based adaptation methodology consisting of four main elements: 1) an adaptation schema (the methodology's backbone) indicating the activities and information flows during the adaptation planning process; 2) a formal model for describing resources and their capabilities; 3) a framework and rules for capability-based matching of product requirements and system capabilities; 4) a preliminary approach to evaluating the impact of changes in product requirements on a production system. These form a comprehensive framework, which tries to address all the facets of the adaptation problem. The emphasis of the work was placed on the development of the resource description and capability model to support the adaptation of a production system. Resource information is regarded as the most important information when making adaptation decisions.

As stated already in the introductory chapter, information which enables adaptation planning and supports adaptation decisions must be gathered from multiple sources. It must be obtained from the experts involved in the processes, such as system integrators, operators and maintenance providers, and from the information systems in which the information is either generated, stored or processed, such as control and monitoring systems, MES (Manufacturing Execution System) or ERP (Enterprise Resource Planning) and simulation systems. Therefore, multiple sources and contributors for the information must be allowed for. This thesis has culminated in a formal, computer-interpretable model for describing resources and their capabilities. As shown in Chapter 5, the use of a common ontology allows the resource information to be understood and contributed to by many entities, and therefore supports the interoperability of the information across multiple software modules and contributors. Together with the development of semantic knowledge management technologies and monitoring systems, it enhances the real-time information management relating to resources' characteristics, status and lifecycle. The lack of this information was identified as one of the main barriers to adaptation in industry today, often forcing new systems to be built from scratch when the production requirements change.

As stated earlier, adaptation is expected to take place not only at the single production system level, but also over distributed factories and logistic networks. This capability model is not restricted to describing production resources in single factories, even though that was the environment for the case studies in this thesis. Instead, the model can also be applied to distributed environments. The capability model could, for example, be used for finding suitable capabilities from a whole supply chain and for dynamically distributing orders within a whole supplier network based on the available capabilities.

During the study, the initial assumption that humans cannot be removed from the adaptation planning process was confirmed. In order to operate in a changing and complex environment, where the quantity and quality of the input information varies case by case, human intelligence should not be replaced by computers. Instead, both should be utilized in appropriate situations. Computers are good at handling, processing and filtering large amounts of data, whereas humans are capable of making intelligent decisions, which often require tacit, experience-based knowledge not managed by computers. Therefore, this thesis proposed a computer-aided, capability-based adaptation methodology in which the computer's processing power is used to generate alternative configuration scenarios for the given product requirements from a large amount of resource information. The human expert can then use his/her intelligence to check the feasibility of the proposed scenarios and to select the best one based on the specified criteria.

The discussion in Chapter 6 evaluated the strengths and limitations of the proposed methodology from multiple perspectives. It gave a thorough description of how each of the developed elements contributes towards the easier adaptation of production systems in a changing environment. In summary, it can be stated that the developed methodology supports adaptation planning and reactive adaptation in the following ways. The adaptation schema provides understanding of the adaptation planning process, its activities and related information flows, and the needed resources, and how these interact with each other. The formal resource model and capability-matching framework facilitate automatic filtering and reasoning with a vast amount of information and the generation of alternative solution scenarios. As a result, less manual information handling and reasoning is required. The resource model provides up-to-date information about the resources to facilitate more reliable adaptation plans. In addition, tools to manage the resource information and input it for the reasoning activities are provided. The developed compatibility domain approach facilitates the comparison between different product concepts in regard to the effort needed to adapt an existing system to new requirements. Consequently, compared to the current manual adaptation practices, less human effort is needed in the adaptation planning process, the adaptation scenarios can be created faster taking all the existing resources into account, and the decisions may be made based on more reliable input data. All of this can result in cost and time savings in the adaptation planning process. Savings can also be expected on the investment side, because the developed methodology encourages adapting and re-using the existing system.

REFERENCES

Al-Safi, Y. & Vyatkin, V., 2007. An ontology-based reconfiguration agent for intelligent mechatronic systems. In: Marik, V., Vyatkin, V. & Colombo, A. (eds.), *Holonic and Multi-Agent Systems for Manufacturing*. Springer-Verlag, pp. 114-126, ISBN 978-3-540-74478-8.

Ameri, F. & Dutta, D., 2008. A Matchmaking Methodology for Supply Chain Deployment in Distributed Manufacturing Environments. *Journal of Computing and Information Science in Engineering*, 8(1), p.9, ISSN 1530-9827.

Avgoustinov, N., 2007. *Modelling in mechanical engineering and mechatronics: towards autonomous intelligent software models*, Springer Verlag, p. 226, ISBN 978-1-84628-908-8.

Barata, J., 2006. The Cobasa Architecture as an Answer to Shop Floor Agility. In: Kordic, V., Lazinica, A. & Merdan, M. (eds.), *Manufacturing the Future*. InTech, pp. 31-76, ISBN 3-86611-198-3.

Barata, J., Camarinhamatos, L. & Candido, G., 2008. A multiagent-based control system applied to an educational shop floor. *Robotics and Computer-Integrated Manufacturing*, 24(5), pp.597-605, ISSN 0736-5845.

Bengel, M., 2007. Modelling objects for skill-based reconfigurable machines. In *Innovative production machines and systems, 3rd I*PROMS Virtual International Conference 2007,* p. 13, ISBN 1-904445-52-7.

Bi, Z.M. & Zhang, W.J., 2001a. Modularity technology in manufacturing: taxonomy and issues. *The International Journal of Advanced Manufacturing Technology*, 18(5), pp. 381–390, ISSN 0268-3768.

Bi, Z.M. & Zhang, W.J., 2001b. Concurrent optimal design of modular robotic configuration. *Journal of Robotic Systems*, 18(2), pp. 77-87, ISSN 0741-2223.

Bi, Z.M., Lang, S.Y.T., Shen, W. & Wang, L., 2008. Reconfigurable manufacturing systems: the state of the art. *International Journal of Production Research*, 46(4), pp. 967-992, ISSN 0020-7543.

Blessing, L.T.M. & Chakrabarti, A., 2009. *DRM, a Design Research Methodology*. London: Springer London, p. 397, ISBN 978-1-84882-586-4.

Boothroyd, G., Dewhurst, P. & Knight, W., 2002. Product Design for Manufacture and Assembly. 2nd edition, CRC Press, p. 720, ISBN 082470584X.

Bourgine, P. & Johnson, J. (Eds.), 2006. *Living roadmap for complex systems science*, Report. ONCE-CS, Open Network of Centres of Excellence in Complex Systems, 71p.

Brehmer, N. & Wang, C., 1999. Reconfigurable manufacturing systems and environment consciousness. In *Proceedings of EcoDesign '99, First international symposium on environmentally conscious design and inverse manufacturing*, pp. 463-468, ISBN 0-7695-0007-2.

Buchanan, B.G. & Duda, R.O., 1983. Principles of rule-based expert systems. *Advances in Computers*, 22, pp. 163-216, ISBN 0-12-012122-0.

Cândido, G. & Barata, J., 2007. A multiagent control system for shop floor assembly. In: Marik, A., Vyatkin, V. & Colombo, V. (eds.), *Holonic and Multi-Agent Systems for Manufacturing*. Heidelberg: Springer, pp. 293–302, ISBN 3-540-74478-9.

CO2PE!, 2010. CO2PE! - Taxonomy. Available at: http://www.mech.kuleuven.be/co2pe!/taxonomy.php [Accessed February 5, 2011].

Colledani, M., Terkaj, W. & Tolio, T., 2009. Product-Process-System Information Formalization. In: Tolio, T. (ed.), *Desing of Flexible Production Systems - Methodologies and Tools*. Springer Berlin Heidelberg, pp. 63-86, ISBN 978-3-540-85413-5.

Dashchenko, A.I. (ed.), 2006. *Reconfigurable Manufacturing Systems and Transformable Factories*. Springer Verlag Berlin Heidelberg, p. 789, ISBN 3-540-29391-4.

Dashchenko, O. A., 2006. Computer Supported Decision Making System that Generates and Optimizes Layouts of Reconfigurable Manufacturing Equipment at the Early Stage of Design. In: Dashchenko, A.I. (Ed.). *Reconfigurable Manufacturing Systems and Transformable Factories*, Springer, pp. 295-325, ISBN 3-540-29391-4.

Deif, A.M. & ElMaraghy, W.H., 2006. A Systematic Design Approach for Reconfigurable Manufacturing Systems. In: ElMaraghy, H.A. & Elmaraghy, W.H. (eds.), *Advances in Design*. Springer, pp. 219-228, ISBN 1-84628-004-4.

Eckert, C., Clarkson, P.J. & Zanker, W., 2004. Change and customisation in complex engineering domains. *Research in Engineering Design*, 15(1), pp. 1-21, ISSN 0934-9839.

ElMaraghy, H. A., 2006. Flexible and reconfigurable manufacturing systems paradigms. *International Journal of Flexible Manufacturing Systems*, 17(4), pp. 261-276, ISSN 0920-6299.

ElMaraghy, H.A. (ed.), 2009. *Changeable and Reconfigurable Manufacturing Systems.* London: Springer London, p. 405, ISBN 978-1-84882-066-1.

EUPASS, 2009. EASE - Evolvable assembly system environment. Available at: http://titanic.uninova.pt/~eupass/index.php?option=com_content&view=category&layout=blog&id=44&Itemid=28 [Accessed September 30, 2010].

Ferreira, P., Lohse, N. & Ratchev, S., 2010. Multi-agent Architecture for Reconfiguration of Precision Modular Assembly Systems. In: Ratchev, S. (ed.), *Precision Assembly Technologies and Systems*. Springer, pp. 247–254, ISBN 3-642-11597-7.

Fleschutz, T., Harms, R., Seliger, G., Bottero, F. & Rusina, F., 2008. Evaluaton of the Reconfiguration and Reuse of Assemlby Equipment. In *Proceedings of 2nd CIRP Conference on Assembly Technology and Systems (CATS2008)*. Toronto.

Fleschutz, T., Harms, R. & Seliger, G., 2009. Valuation of Assembly Equipment Reuse with Real Options. In *20th Annual Conference of the Production and Operations Management Society*.

Frei, R., Ferreira, B., Di Marzo Serugendo, G. & Barata, J., 2009. An architecture for self-managing evolvable assembly systems. In *IEEE International Conference on Systems, Man and Cybernetics*. San Antonio, Texas, pp. 2707-2712, ISSN 1062-922X.

Frei, R., Di Marzo Serugendo, G., Pereira, N., Belo, J. & Barata, J., 2010. Implementing self-organisation and self-management in evolvable assembly systems. In *2010 IEEE International Symposium on Industrial Electronics*. IEEE, pp. 3527-3532, ISBN 978-1-4244-6390-9.

Frei, R., 2010. *Self-organization in evolvable assembly systems*. PhD thesis. University of Lisbon., p. 273,.

Frei, R. & Di Marzo Serugendo, G., 2011. Self-Organizing Assembly Systems. *Systems, Man, and Cybernetics,* 41(6), pp. 885-897, ISSN 1094-6977.

Fryer, P., 2010. A brief description of Complex Adaptive Systems and Complexity Theory. Available at: http://www.trojanmice.com/articles/complexadaptivesystems.htm [Accessed May 25, 2010].

Garcia, F., Lanz, M, Luostarinen, P. & Andersson, P.H., 2010. Feature-based unfolding algorithm for sheet metal products. In *5th Americas International Conference on Production Research, ICPR 2010*. p. 5, ISBN 978-958-44-7045-4.

Garcia, F., Lanz, M., Järvenpää, E. & Tuokko, R., 2011. Process planning based on feature recognition method. In *IEEE International Symposium on Assembly and Manufacturing (ISAM)*, p. 5, ISBN 978-1-61284-342-1.

Gen, M. & Cheng, R., 1997. *Genetic Algorithms & Engineering Design*, New York: John Wiley & Sons, Inc., p. 411, ISBN 0-471-12741-8.

Gero, J.S., 1990. Design prototypes: a knowledge representation schema for design. *AI magazine*, 11(4), pp. 26-36.

Giret, A. & Botti, V., 2004. Holons and agents. *Journal of Intelligent Manufacturing*, 15(5), pp. 645-659, ISSN 0956-5515.

Grabowik, C. & Knosala, R., 2003. The method of knowledge representation for a CAPP system. *Journal of Materials Processing Technology*, 133(1-2), pp. 90–98, ISSN 0924-0136.

Gruber, T., 1993. A translation approach to portable ontology specifications. *Knowledge acquisition*, 5(2), pp. 199-220.

Harms, R., Fleschutz, T. & Seliger, G., 2008. Knowledge Based Approach to Assembly System Reuse. In *ASME 2008 9th Biennial Conference on Engineering Systems Design and Analysis*, pp. 295-302, ISBN 978-0-7918-4835-7.

Harms, R., Fleschutz, T. & Seliger, G., 2009. Evaluation and Planning of Assembly Equipment Applying Knowledge Based Systems. In *20th Annual Conference of the Production and Operations Management Society*.

Hedlind, M, Lundgren, M & Lundholm, T., 2010. Model based machining descriptions. In *CIRP Intelligent Computation in Manufacturing Engineering,* 4 p., ISBN 978-88-95028-65-1.

Hedlind, M., Lundgren, M. & Archenti, A., 2010. Manufacturing resource modelling for model driven operation planning. In *CIRP 2nd International Conference on Process Machine Interactions, Canada,* 10p, ISBN 978-0-9866331-0-2.

Hirani, H., Ratchev, S., Lohse, N. & Valtchanov, G., 2006. Methodology for knowledge enriched requirements specification for assembly system reconfiguration. *Assembly Automation*, 26(4), pp. 307-314, ISSN 0144-5154.

Ho, J.K.L., 2005. A proposed approach for reconfiguration of flexible assembly line systems by motion genes. *International Journal of Production Research*, 43(9), pp. 1729-1749, ISSN 0020-7543.

Holmström, P., 2006. *Modelling Manufacturing Systems Capability*. Lic. thesis. Royal Institute of Technology, Stocholm, Sweden., p. 73, ISSN 1650-1888.

Hong, N.K. & Hong, S., 1998. Entity-based models for computer-aided design systems. *Journal of computing in civil engineering*, pp. 30-41, ISSN 0887-3801.

Hopgood, A.A., 2001. *Intelligent Systems for Engineers and Scientists*, p. 461, ISBN 0849304563.

Jackson, M.C., 2003. *Systems thinking: Creative holism for managers*, Wiley, Chichester, UK, p. 376, ISBN 0470845228.

Jang, S.-H., Jung, Y.-M., Hwang, H.-Y., Choi, Y.-H. & Park, J.-K. 2008. Development of Reconfigurable Micro Machine Tool for Microfactory. *International Conference on Smart Manufacturing Application*, pp. 190-195, ISBN 978-89-950038-8-6.

Jervis, R., 1998. *System effects: Complexity in political and social life*, University Press, Princeton, p. 328, ISBN 9781400822409.

Jovane, F., Westkämper, E. & Williams, D., 2009. *The ManuFuture Road – Towards Competitve and Sustainable High-Adding-Value Manufacturing*, Springer, p. 261, ISBN 978-3-540-77011-4.

Järvenpää, E., Lanz, M., M., J. & Tuokko, R., 2010. Studying the Information Sources and Flows in a Company – Support for the Development of New Intelligent Systems. In *Flexible Automation and Intelligent Manufacturing, FAIM 2010*. California. p. 8.

Järvenpää, E., Luostarinen, P., Lanz, M., & Tuokko, R., 2011a. Presenting capabilities of resources and resource combinations to support production system adaptation. In *IEEE International Symposium on Assembly and Manufacturing (ISAM)*, p. 6, ISBN 978-1-61284-342-1.

Järvenpää, E., Luostarinen, P., Lanz, M., Garcia, F. & Tuokko, R., 2011b. Dynamic operation environment — Towards intelligent adaptive production systems. In *IEEE International Symposium on Assembly and Manufacturing (ISAM)*, p. 6, ISBN 978-1-61284-342-1.

Järvenpää, E., Luostarinen, P., Lanz, M. & Tuokko, R., 2012a. Adaptation of Manufacturing Systems in Dynamic Environment Based on Capability Description Method. In F. Abdul Aziz (ed.), *Manufacturing System*. InTech, p. 26, ISBN 978-953-51-0530-5.

Järvenpää, E., Luostarinen, P., Lanz, M. & Tuokko, R. 2012b. Development of a Rule-base for Matching Product Requirements against Resource Capabilities in an Adaptive Production System. In *Flexible Automation and Intelligent Manufacturing, FAIM 2012*, Helsinki, Finland, p. 8.

Kitamura, Y., Koji, Y. & Mizoguchi, R., 2006. An ontological model of device function: industrial deployment and lessons learned. *Applied Ontology*, 1(3-3), pp. 237–262, ISNN 1570-5838.

Kitamura, Y. & Mizoguchi, R., 2010. Characterizing functions based on ontological models from an engineering point of view. In *Proceeding of the 6th international conference on Formal Ontology in Information Systems (FOIS 2010)*. IOS Press, pp. 301–314, ISBN 978-1-60750-534-1.

Kjellberg, T., von Euler-Chelpin, A., Hedlind, M., Lundgren, M., Sivard, G. & Chen, D., 2009. The machine tool model—A core part of the digital factory. *CIRP Annals - Manufacturing Technology*, 58(1), pp. 425-428, ISSN 0007-8506.

Knowledge Based Systems Inc., 2010. IDEF0 Function Modeling Method. Available at: http://www.idef.com/IDEF0.htm [Accessed November 1, 2010].

Koestler, A., 1967. *The Ghost in the Machine*, Arkana Books, London.

Koren, Y., Heisel, U., Jovane, F., Moriwaki, T., Pritschow, G., Ulsoy, G. & Van Brussel, H., 1999. Reconfigurable Manufacturing Systems. *CIRP Annals - Manufacturing Technology*, 48(2), pp. 527-540, ISSN 0007-8506.

Koren, Y., 2006. General RMS Characteristics, Comparison with Dedicated and Flexible Systems. In: Dashchenko, A.I. (ed.), *Reconfigurable Manufacturing Systems and Transformable Factories*. Springer, pp. 27-46, ISBN 3-540-29391-4.

Koren, Y. & Shpitalni, M., 2010. Design of reconfigurable manufacturing systems. *Journal of Manufacturing Systems*, 29(4), pp. 130-141, ISBN 0278-6125.

Landers, R.G., Ruan, J. & Liou, F., 2006. Reconfigurable Manufacturing Equipment. In A. I. Daschenko, ed. *Reconfigurable Manufacturing Systems and Transformable Factories*. Sringer, pp. 79-110, ISBN 3-540-29391-4.

Lanz, M., Kallela, T., Velez, G. & Tuokko, R., 2008. Product, process and system ontologies and knowledge base for managing knowledge between different clients. In *2008 IEEE International Conference on Distributed Human-Machine Systems*, pp. 508-513, ISBN 978-80-01-04028-7.

Lanz, M., Rodriguez, R., Luostarinen, P. & Tuokko, R., 2010a. Neutral interface for assembly and manufacturing related knowledge exchange in heterogeneous design environment. In: Ratchev, S. (ed.), *Precision Assembly Technologies and Systems*. Springer, pp. 21–29, ISBN 978-3-642-11597-4.

Lanz, M, Nylund, H & Ranta, A., 2010b. Set-up and first steps on capturing of realistic resource characteristics of an intelligent manufacturing environment. In *Proceedings of the 20th International Conference on Flexible Automation and Intelligent Manufacturing, FAIM 2010*. pp. 1028-1036.

Lanz, M., 2010. *Logical and Semantic Foundations of Knowledge Representation for Assembly and Manufacturing Processes*. PhD thesis. Tampere University of Technology, p. 138, ISBN 9789521523939.

Lanz, M, Järvenpää, E, Luostarinen, P, Tenhunen, A., Tuokko, R & Rodriguez, R, 2011. Formalizing Connections between Products , Processes and Resources – Towards Semantic Knowledge Management System. In *The 4th International Swedish Production Symposium SPS11*. Lund, pp. 425-431.

Lanz, M., Järvenpää, E., Garcia, F., Luostarinen, P. & Tuokko, R., 2012. Towards Adaptive Manufacturing Systems - Knowledge and Knowledge Management Systems. In Abdul Aziz, F. (ed.), *Manufacturing System*. InTech, p. 18, ISBN 978-953-51-0530-5.

Lehtonen, T., 2007. *Designing Modular Product Architecture in the New Product Development*. PhD thesis. Tampere University of Technology, p. 220, ISBN 9789521518980.

Levin, M., 2002. Towards combinatorial analysis, adaptation, and planning of human-computer systems. *Applied Intelligence*, 16, pp. 235-247, ISSN 0924-669X.

De Lit, P. & Delchambre, A., 2003. *Integrated design of a product family and its assembly system*, Kluwer Academich Publishers, p. 281, ISBN 1-4020-7437-9.

Lohse, N., 2006. *Towards an ontology framework for the integrated design of modular assembly systems*. PhD thesis. University of Nottingham, p. 245, ISBN 0387312765.

Lohse, N., Hirani, H. & Ratchev, S., 2006a. Equipment ontology for modular reconfigurable assembly systems. *International Journal of Flexible Manufacturing Systems*, 17(4), pp. 301-314, ISSN 0920-6299.

Lohse, N., Schäfer, C. & Ratchev, S., 2006b. Towards an Integrated Assembly Process Decomposition and Modular Equipment Configuration. In: Ratchev, S. (ed.), *Precision Assembly Technologies for Mini and Micro Products*, Springer, pp. 215–225, ISBN 978-0-387-31276-7.

Lohse, N., Maraldo, T. & Barata, J., 2008. EUPASS std-0007: Assembling Process Ontology Specification. 38 p.

Malec, J., Nilsson, A., Nilsson, K. & Nowaczyk, S., 2007. Knowledge-based reconfiguration of automation systems. In *3rd Annual IEEE Conference on Automation Science and Engineering*, pp. 170–175, ISBN 978-1-4244-1154-2.

Marler, R.T. & Arora, J.S., 2004. Survey of multi-objective optimization methods for engineering. *Structural and Multidisciplinary Optimization*, 26(6), pp. 369-395, ISSN 1615-147X.

Maropoulos, P.G., McKay, K.R. & Bramall, D.G., 2002. Resource-Aware Aggregate Planning for the Distributed Manufacturing Enterprise. *CIRP Annals - Manufacturing Technology*, 51(1), pp. 363-366, ISSN 0007-8506.

Maropoulos, P G, Bramall, D G, McKay, K R, Rogers, B. & Chapman, P., 2003. An aggregate resource model for the provision of dynamic “resource-aware” planning. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, 217(10), pp. 1471-1480, ISSN 0954-4054.

Martinez Lastra, J.L., 2004. *Reference Mechatronic Architecture for Actor-based Assembly Systems*. PhD thesis. Tampere University of Technology, p. 112, ISBN 952-15-1210-5.

Martines Lastra, J.L., Insaurralde, C. & Colombo, A., 2009. Agent-based control for desktop assembly factories. In: Wang, L. & Nee, A. Y. C. (eds.), *Collaborative design and planning for digital manufacturing*. Springer, pp. 265-291, ISBN 978-1-84882-286-3.

Mehrabi, M.G., Ulsoy, A.G. & Koren, Y., 2000. Reconfigurable manufacturing systems: key to future manufacturing. *Journal of Intelligent Manufacturing*, 11(4), pp. 403–419, ISSN 0956-5515.

Minhas, S. & Berger, U., 2011. A reconfiguration concept to enable versatile production in the automotive factories. In: ElMaraghy, H.A. (ed.), *Enabling Manufacturing Competitiveness and Economic Sustainability*. Springer, pp. 352–357, ISBN 978-3-642-23860-4.

Monostori, L., Váncza, J. & Kumara, S.R.T., 2006. Agent-Based Systems for Manufacturing. *CIRP Annals - Manufacturing Technology*, 55(2), pp. 697-720, ISSN 0007-8506.

Moon, Y.M., 2006. Reconfigurable Machine Tool Design. In: Dashchenko, A.I. (Ed.). *Reconfigurable Manufacturing Systems and Transformable Factories*, Springer, pp. 111-139, ISBN 3-540-29391-4.

Nylund, H., Salminen, K. & Andersson, P., 2008. Digital Virtual Holons – An Approach to Digital Manufacturing Systems. In: Mitsuishi, M., Ueda, K. & Kumura, F. (eds.), *Manufacturing Systems and Technologies for the New Frontier*. Springer, pp. 103–106, ISBN 978-1-84800-266-1.

Nylund, H, Salminen, K & Andersson, PH, 2011. Framework for Distributed Manufacturing Systems. In ElMaraghy, H.A. (ed.), *Enabling Manufacturing Competitiveness and Economic Sustainability*. Springer, pp. 172–177, ISBN 978-3-642-23860-4.

Oduguwa, P. a, Roy, R. & Sackett, P.J., 2006. Cost impact analysis of requirement changes in the automotive industry: a case study. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, 220(9), pp. 1509-1525, ISBN 0954-4054.

Onori, M, Neves, P., Akillioglu, H. & Maffei, A., 2011. Dealing with the unpredictable: An Evolvable Robotic Assembly Cell. In; ElMaraghy, H.A. (ed.), *Enabling Manufacturing Competitiveness and Economic Sustainability*. Springer, pp. 160-165, ISBN 978-3-642-23860-4.

Onori, M., Semere, D. & Lindberg, B., 2010. Evolvable systems: an approach to self-X production. In: Huang, G., Mak, K. & Maropoulos, P. (eds.), *Proceedings of the 6th CIRP-Sponsored International Conference on Digital Enterprise Technology*. Springer Berlin / Heidelberg, pp. 789-802, ISBN 978-3-642-10429-9.

Pahl, G. & Beitz, W., 1996. *Engineering Design - A Systematic Approach* 2nd edition., Springer-Verlag, Berlin/Heidelberg, p. 544, ISBN 3540199179.

Perremans, P., 1996. Feature-based description of modular fixturing elements: The key to an expert system for the automatic design of the physical fixture. *Advances in Engineering Software*, 25(1), pp. 19-27, ISSN 09659978.

Rahimifard, A. & Weston, R.H., 2009. A resource-based modelling approach to support responsive manufacturing systems. *The International Journal of Advanced Manufacturing Technology*, 45(11-12), pp. 1197-1214, ISSN 0268-3768.

Ramage, M. & Shipp, K., 2009. *Systems Thinkers*, Springer Verlag, London, p. 316, ISBN 1848825242.

Rampersad, H.K., 1994. *Integrated and Simultaneous Design for Robotic Assembly*, Chichester: John Wiley & Sons Ltd., p. 212, ISBN 0471954667.

Ratchev, S., Hirani, H. & Bonney, M., 2007. Knowledge based formation of re-configurable assembly cells. *Journal of Intelligent Manufacturing*, 18(3), pp. 401-409, ISSN 0956-5515.

Reinhart, G & Meling, F., 2011. Methodology for the Support of Reconfiguration Processes in the Automotive Body Shop. In: ElMaraghy, H.A. (ed.), *Enabling Manufacturing Competitiveness and Economic Sustainability*. Springer, pp. 358–363, ISBN 978-3-642-23860-4.

Rekiek, B., Delchambre, A. & Dolgui, A., 2002. Assembly line design: a survey. Proceedings of the 15th *IFAC World Congress*.

SIARAS, 2008. *SIARAS: Skill-based Inspection and Assembly for Reconfigurable Automation Systems*, Report. 17 p.

Saarinen, E. & Hämäläinen, R.P., 2004. Systems intelligence: Connecting engineering thinking with human sensitivity. In: Hämäläinen, R.P. & Saarinen, E. (eds.), *Systems intelligence: Discovering a hidden competence in human action and organizational life*. Helsinki University of Technology, Systems Analysis Laboratory, Research Reports A88, pp. 9-38, ISBN 951-22-7168-0.

Saarinen, E. & Hämäläinen, R.P., 2010. The Originality of Systems Intelligence. In: Hämäläinen, R.P. & Saarinen, E. (eds.), *Essays on Systems Intelligence*. Aalto University, School of Science and Technology, Systems Analysis Laboratory, pp. 9-26.

Salminen, K., Nylund, H. & Andersson, P.H., 2009. Role based self-adaptation of a robot DiMS based on system intelligence approach. In *19th International Conference on Flexible Automation and Intelligent Manufacturing, (FAIM)*, pp. 964-970, ISBN 978-0-9562303-2-4.

Semere, D., Onori, M., Maffei, A. & Adamietz, R., 2008. Evolvable assembly systems: coping with variations through evolution. *Assembly Automation*, 28(2), pp. 126-133, ISSN 0144-5154.

Senge, P.M., 1990. *The fifth discipline*, Doubleday, New York, p. 424, ISBN 0385260954.

Siltala, N., Hoffmann, A., Gengenbach, U., Traivani, E. & Vollmer, H., 2008. EUPASS std-0004: Emplacement Specification and Guideline. 61 p.

Siltala, N., Hofmann, A., Tuokko, R. & Bretthauer, G., 2009. Emplacement and blue print - An approach to handle and describe modules for evolvable assembly systems. In *9th IFAC Symposium on Robot Control, SYROCO '09*. pp. 183-188.

Siltala, N. & Tuokko, R., 2010. Emplacement and Blue Print–Electronic Module Description Supporting Evolvable Assembly Systems Design, Deployment and Execution. In: Huang, G.Q., Mak, K.L. & Maropoulos, P.G. (eds.), *Proceedings of the 6th CIRP-Sponsored International Conference on Digital Enterprise Technology*. Berlin, Heidelberg: Springer, pp. 773–788, ISBN 978-3-642-10429-9.

Slack, N., Chambers, S. & Johnston, R., 2004. *Operations Management* 4th ed., Prentice Hall, p. 794, ISBN 0-273-67906-6.

Slama,S., 2002. The importance of modularity – Batch-size flexible assembly systems. Product Life Cycle Management & Easier Automation, Paterspand, Turnhout, Belgium.

Smale, D. & Ratchev, 2009. A Capability Model and Taxonomy for Multiple Assembly System Reconfigurations. In B. Natalia (ed.), In *13th IFAC Symposium on Information Control Problems in Manufacturing*. Moscow, pp. 1923-1928.

Smale, D. & Ratchev, S., 2010. Application of a Reconfiguration Methodology for Multiple Assembly System Reconfigurations. In: Ratchev, S. (ed.), *Precision Assembly Technologies and Systems*. Springer Boston, pp. 239-246, ISBN 978-3-642-11597-4.

Stanford Center for Biomedical Informatics Research, 2012. Protégé ontology editor. Available at: http://protege.stanford.edu/ [Accessed March 1, 2012].

Suh, N.P., 1998. Engineering Design Axiomatic Design Theory for Systems. *Research in Engineering Design*, 10, pp. 189-209, ISSN 0934-9839.

Tang, L., 2005. *Design and Reconfiguration of RMS for Part Family*. PhD thesis. University of Michigan, p. 164.

Terkaj, W., Tolio, T. & Valente, A. 2009a. A Review on Manufacturing Flexibility. In: Tolio, T. (ed.), *Desing of Flexible Production Systems - Methodologies and Tools*. Springer Berlin Heidelberg, pp. 41-61, ISBN 978-3-540-85413-5.

Terkaj, W., Tolio, T. & Valente, A. 2009b. Design of Focused Flexibility Manufacturing Systems (FFMSs). In: Tolio, T. (ed.), *Desing of Flexible Production Systems - Methodologies and Tools*. Springer Berlin Heidelberg, pp. 137-190, ISBN 978-3-540-85413-5.

Terwiesch, C. & Loch, C.H., 1999. Managing the Process of Engineering Change Orders: The Case of the Climate Control System in Automobile Development. *Journal of Product Innovation Management*, 16(2), pp. 160-172, ISSN 0737-6782.

Tharumarajah, A., Wells, A.J. & Nemes, L., 1998. Comparison of emerging manufacturing concepts. *IEEE International Conference on Systems, Man, and Cybernetics, 1998*, 1, pp. 325-331, ISSN 1062-922X.

Timm, I. & Woelk, P.-O., 2003. Ontology-based capability management for distributed problem solving in the manufacturing domain. In: Schillo, M., Klusch, M., Müller, J. & Tianfield, H. (eds.), *Multiagent System Technologies*. Springer Berlin / Heidelberg, pp. 168-179, ISBN 978-3-540-20124-3.

Timm, I.J., Scholz, T. & Herzog, O., 2006. Capability-based emerging organization of autonomous agents for flexible production control. *Advanced Engineering Informatics*, 20(3), pp. 247-259, ISSN 1474-0346.

Tolio, T. & Valente, A., 2006. An Approach to Design the Flexibility Degree in Flexible Manufacturing Systems. In *16th Int. Conf. on Flexible Automation and Intelligent Manufacturing, FAIM 2006, June 26 – 28*. Limerick, pp. 1229-1236.

Tolio, T, Ceglarek, D., ElMaraghy, H.A., Fischer, A., Hu, S., Laperriere, L., Newman, S. & Vancza, J. (2010). SPECIES - Co-evolution of Products, Processes and Production Systems. *CIRP ANNALS – Manufacturing Technology*, 59(2), p. 672-693, ISSN 0007-8506.

Tracht, K. & Hogreve, S., 2011. Decision Making During Design and Reconfiguration of Modular Assembly Lines. In: ElMaraghy, H.A. (ed.), *Enabling Manufacturing Competitiveness and Economic Sustainability*. Springer, pp. 105-110, ISBN 978-3-642-23860-4.

Travaini, E., Pedrazzoli, P., Rinaldi, R. & Boër, C.R., 2002. Methodological Approach and Reconfiguration Tool for Assembly Systems. *CIRP Annals - Manufacturing Technology*, 51(1), pp. 9-13, ISSN 00078506.

Tsai, Y.-T. & Wang, K.-S., 1999. The development of modular-based design in considering technology complexity. *European Journal of Operational Research*, 119(3), pp. 692-703, ISSN 0377-2217.

Ueda, K., Vaario, J. & Ohkura, K., 1997. Modelling of biological manufacturing systems for dynamic reconfiguration. *CIRP Annals-Manufacturing Technology*, 46(1), pp. 343-346, ISSN 0007-8506.

Ueda, K, Markus, A., Monostori, L., Kals, H. & Arai, T., 2001. Emergent Synthesis Methodologies for Manufacturing. *CIRP Annals - Manufacturing Technology*, 50(2), pp. 535-551, ISSN 0007-8506.

Ueda, K., 2007. Emergent Synthesis Approaches to Biological Manufacturing Systems. In: Cunha, P. & Marapoulos, P. (eds.), *Digital Enterprise Technology*. Elsevier, pp. 25-34, ISBN 978-0-387-49864-5.

Usher, J. & Fernandes, K., 1999. An object-oriented application of tool selection in dynamic process planning. *International Journal of Production Research*, 37(13), pp. 2879–2894, ISSN 0020-7543.

Van Brussel, H., Wyns, J., Valckenaers, P., Bongaerts, L. & Peeters, P., 1998. Reference architecture for holonic manufacturing systems: PROSA. *Computers in Industry*, 37(3), pp. 255-274, ISSN 0166-3615.

Vichare, P., Nassehi, A., Kumar, S. & Newman, S.T., 2009. A Unified Manufacturing Resource Model for representing CNC machining systems. *Robotics and Computer-Integrated Manufacturing*, 25(6), pp. 999-1007, ISSN 0736-5845.

Visual Components, 2012. 3DCreate simulation software. Available at: http://www.visualcomponents.com/Products/3DCreate [Accessed February 1, 2012].

von Euler-Chelpin, A. & Kjellberg, T., 2007. Capturing resource operation knowledge from runtime data for production support and feedback to development. In Cunha, P. & Maropoulos, P. (eds.), *Digital Enterprise Technology*. Springer US, pp. 519–526, ISBN 978-0-387-49864-5.

Vos, J.A.W.M., 2001. *Module and System Design in Flexibly Automated Assembly*. PhD thesis. Delft University of Technology, p. 198.

Warnecke, H.J., 1993. *The Fractal Company: A Revolution in Corporate Culture*, Springer, p. 228, ISBN 354056537X.

Watson, I., 1999. Case-based reasoning is a methodology not a technology. *Knowledge-Based Systems*, 12(5-6), pp. 303-308, ISSN 0950-7051.

Westkämper, E., 2006. Factory Transformability: Adapting the Structures of Manufacturing. In Daschenko, A.I. (ed.), *Reconfigurable Manufacturing Systems and Transformable Factories*. Springer Berlin / Heidelberg, pp. 371-381, ISBN 3-540-29391-4.

Wiendahl, H., ElMaraghy, H., Nyhuis, P., Zah, M., Duffie, N. & Brieke, M., 2007. Changeable Manufacturing - Classification, Design and Operation. *CIRP Annals - Manufacturing Technology*, 56(2), pp. 783-809, ISSN 0007-8506.

Wiendahl, H.P. & Heger,, C.L., 2004. Justifying Changeability. A Methodical Approach to Achieving Cost Effectiveness. *Journal for Manufacturing Science and Production*, 6(1-2), pp. 33-40, ISSN 0793-6648.

Wikipedia, 2012. Denavit-Hartenberg parameters. Available at: http://en.wikipedia.org/wiki/Denavit%E2%80%93Hartenberg_parameters [Accessed September 1, 2010].

Wikipedia, 2011. List of manufacturing processes. Available at: http://en.wikipedia.org/wiki/List_of_manufacturing_processes [Accessed February 5, 2011].

Wildemann, H., 2009. *Variant Management: Guidelines for the Reduction, Control and Avoidance of Complexity in Products and Processes (German Title: Variantenmanagement: Leitfaden zur Komplexitätsreduzierung, -beherrschung und -vermeidung in Produkt und Prozess)*, TCW Transfer Centre for Production Logistics and Technology Management, München.

Wänström, C. & Jonsson, P., 2006. The impact of engineering changes on materials planning. *Journal of Manufacturing Technology Management*, 17(5), pp. 561-584, ISSN 1741-038X.

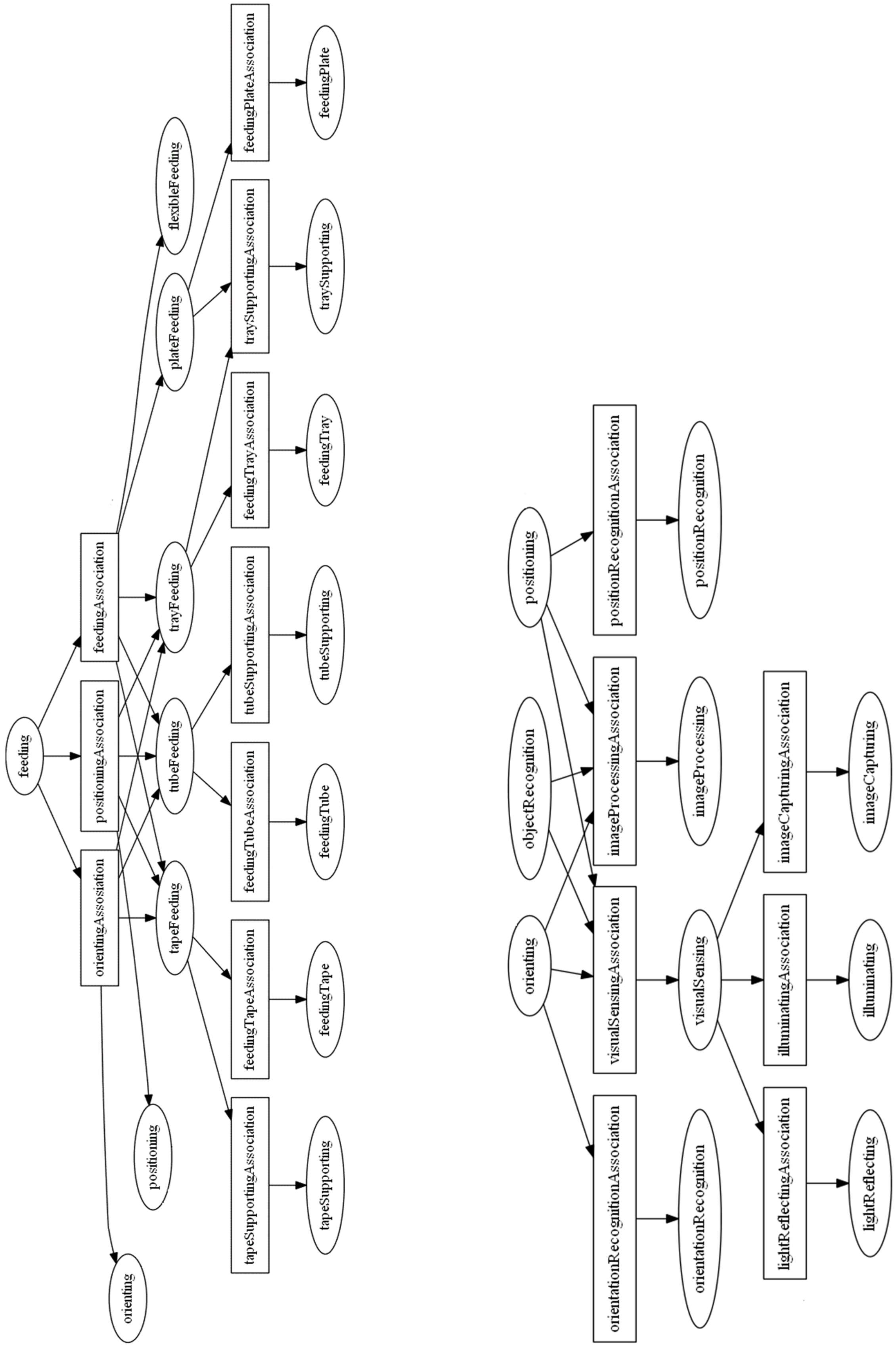
Youssef, A.M.A. & ElMaraghy, H. A., 2007. Optimal configuration selection for Reconfigurable Manufacturing Systems. *International Journal of Flexible Manufacturing Systems*, 19(2), pp. 67-106, ISSN 0920-6299.

Zäh, M.F., Reinhart, G., Ostgathe, M., Geiger, F. & Lau, C., 2010. A holistic approach for the cognitive control of production systems. *Advanced Engineering Informatics*, 24(3), pp. 300-307, ISSN 1474-0346.

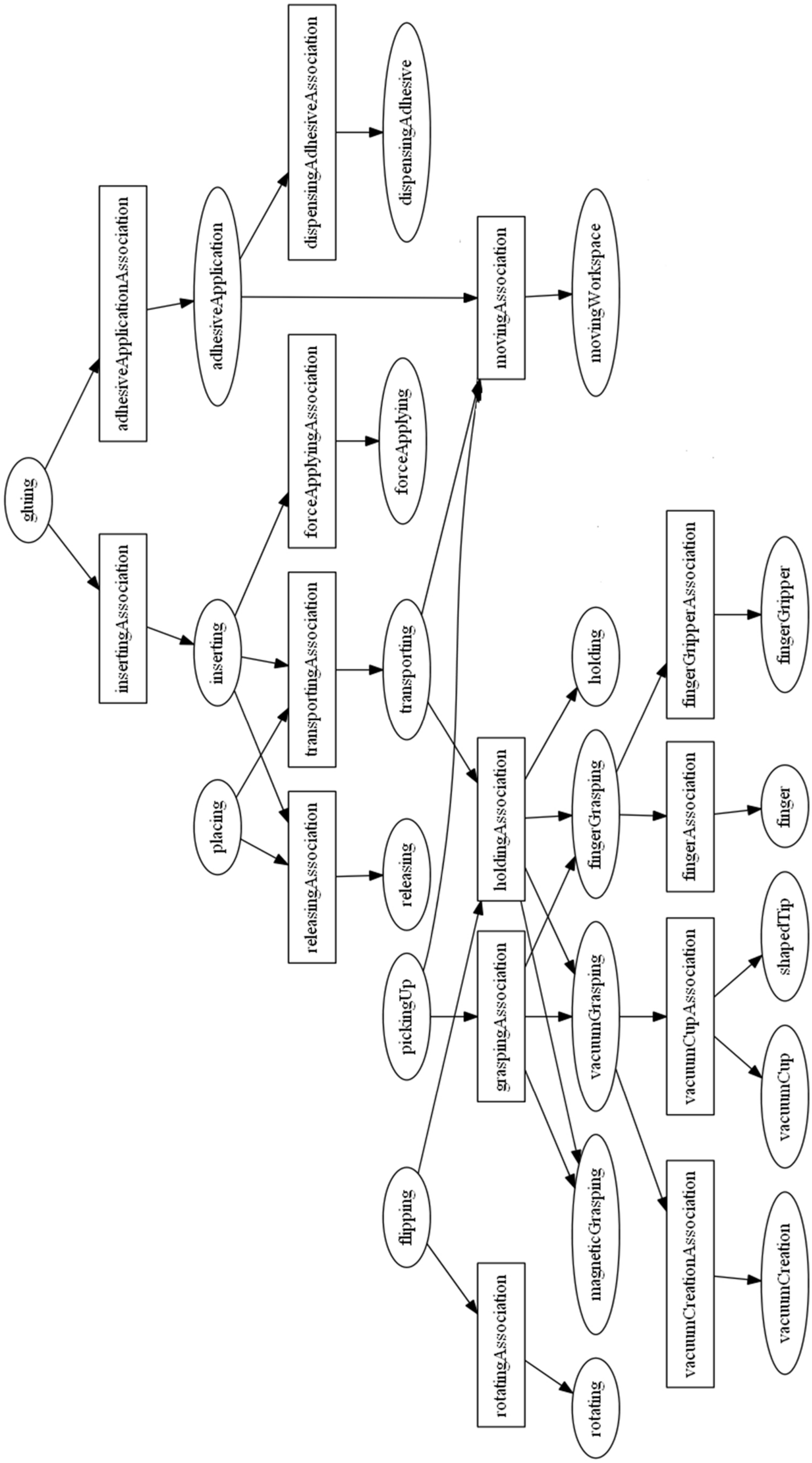
Zäh, M., Ostgathe, M, Geiger, F & Reinhart, G, 2011. Adaptive Job Control in the Cognitive Factory. In: ElMaraghy, H.A. (ed.), *Enabling Manufacturing Competitiveness and Economic Sustainability*. Springer, pp. 10–17, ISBN 978-3-642-23860-4.

APPENDICES

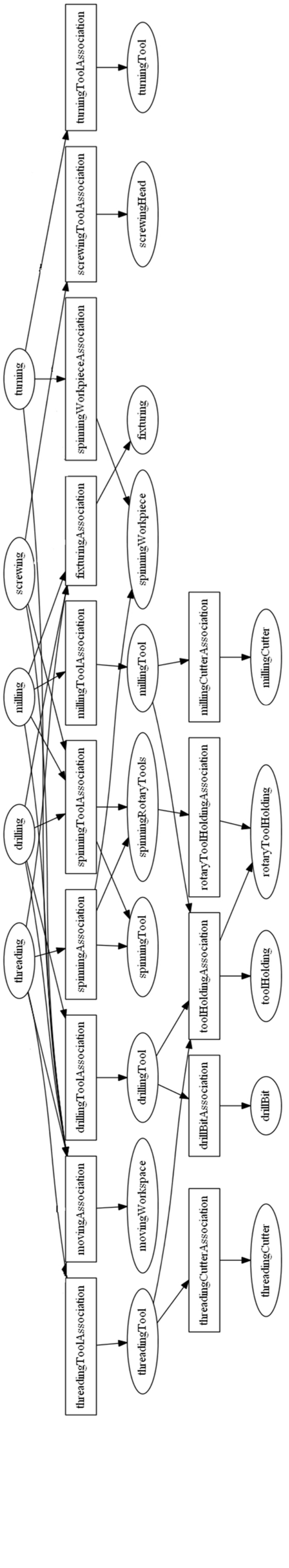
Appendix 1: Capability model – Part 1: Feeding & Object recognition

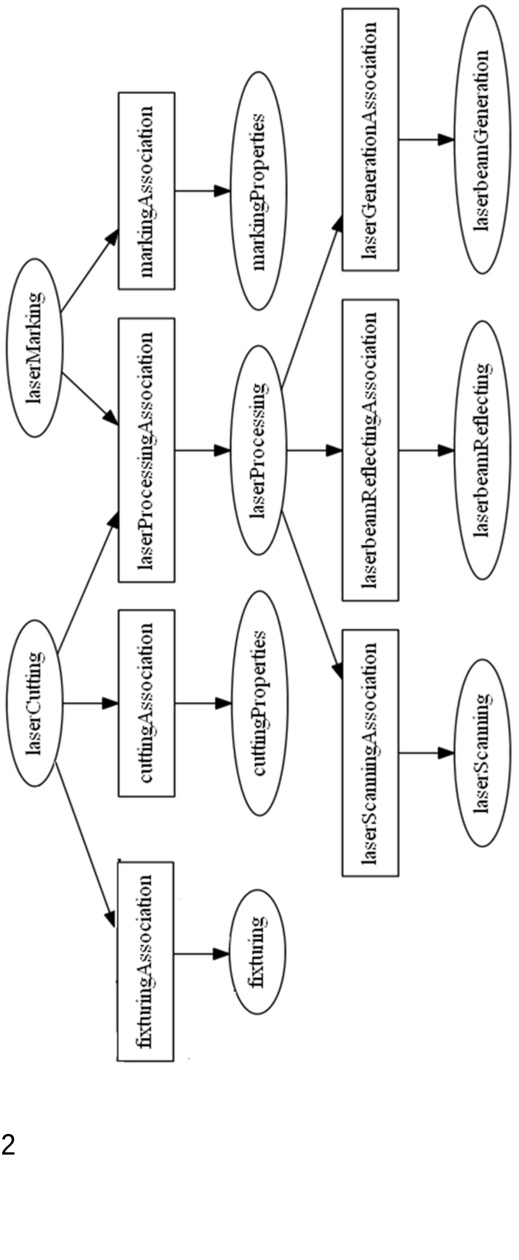


Appendix 2: Capability model – Part 2: Gluing, pick&place, inserting



Appendix 3: Capability model – Part 3: machining & lasering





Appendix 4: Capability taxonomy

Production						
	Manufacturing					
		DiscreteManufacturing				
			Shaping			
				MaterialAdding		
					Casting	
					Moulding	
					LaserAdding	
				Material removing		
					Punching	
					Machining	
						Milling
						Turning
						Drilling
						Boring
						Threading
						Grooving
					Laser cutting	
					Grinding	
				Material conserving		
					Forming	
						Bending
						Pressing
			Non-shaping			
				Surface finishing		
					Coating	
					Painting	
				Heat treatment		
				Marking		
					LaserMarking	
					StampMarking	
					Printing	
					Writing	
					InkMarking	
					Labelling	
		ContinuousManufacturing				
	Assembling					
		Joining				
			DeformingFixating			
				Elastic deforming		
					Pressing	
					Snapping	
					Shrinking	
					Wedging	
					Clipping	
				Plastic deforming		
					Bending	
					Twisting	
					Clinching	
					Crimping	
					Folding	

			Fastening			
				Riveting		
				Screwing		
				Sewing		
				Spring fastening		
				Stapling		
				Taping		
				Nailing		
				Clamping		
			Self sticking			
			Gluing			
			Adhesive curing			
				UV curing		
				Thermal curing		
			Soldering			
				Dip soldering		
				Furnace soldering		
				Induction soldering		
				Iron soldering		
				Resistance soldering		
				Torch soldering		
			Welding			
				Cold welding		
				Metal welding		
					Arc welding	
					Laser welding	
					Resistance welding	
				Plastic welding		
					Friction welding	
					Heated air welding	
					Ultrasonic welding	
	Logistic					
		Manipulation				
			Picking up			
				Grasping		
					FingerGrasping	
					VacuumGrasping	
			Holding			
			Arranging			
				Placing		
				Mating		
				Inserting		
				Orienting		
				Positioning		
			Transporting			
			Flipping			
		Buffering				
		Storing				
		Routing				
		Feeding				
			BulkFeeding			
				Plate feeding (flexible tray)		
				Sword feeding		
				Vibratory feeding		
				Bowl feeding		
				Flexible feeding		
			Tube feeding			
			Tape feeding			
			Tray feeding (dedicated tray)			

	Preparation					
		Loading				
		Fixturing				
		Adhesive application				
			Adhesive dispensing			
			Adhesive printing			
			Self adhesive tape application			
		Solder paste application				
			Solder paste dispensing			
			Solder paste screen printing			
		Unpacking				
		Cleaning				
	Qualifying					
		Identification				
			Scanning			
				RF scanning		
				Laser scanning		
			Vision reading			
				Dotcode reading		
				Barcode reading		
			Image capturing			
		Inspection				
		Testing				
		Measuring				
	Finalization					
		Marking				
			Writing			
			Ink marking			
			Stamp marking			
			Printing			
			Labelling			
			Laser marking			
		Packaging				
			Wrapping			
			Boxing			
			Bagging			
			Sealing			
			Encapsulating			
		Coating				
		Cleaning				
		Unloading				

Appendix 5: Example rules in the rule-base

DOMAIN EXPERT RULES

Detailed capability definition

Taxonomy: MaterialRemoving

```
IF feature.hasCapabilityTaxonomy(“MaterialRemoving”) AND
   feature.hasFeatureType (“hole”) AND
   feature.hasFeatureType(“cylindrical”) AND
   feature.getParameter(“diameter”) <= 40.0
THEN
  RETURN [Capabilities.find(“drilling”)]


IF feature.hasCapabilityTaxonomy(“MaterialRemoving”) AND
   feature.hasFeatureType(“hole”) AND
   feature.hasFeatureType(“cylindrical”) AND
   feature.getParameter(“diameter”) > 40.0
THEN
   RETURN [Capabilities.find(“drilling”), Capabilities.find(“boring”)]


IF feature.hasCapabilityTaxonomy(“MaterialRemoving”) AND
   product.getInitialProduct.hasShape(“cylindrical”) AND
   feature.hasFeatureType(“cylindrical”) AND
   (feature.hasFeatureType(“groove”) OR
    feature.hasFeatureType(“pad”) OR
    feature.hasFeatureType(“chamfer”) OR
    feature.hasFeatureType(“rounding”))
THEN
  RETURN [Capabilities.find(“turning”), Capabilities.find(“milling”)]


IF feature.hasCapabilityTaxonomy(“MaterialRemoving”) AND
   (feature.hasFeatureType(“groove”) OR
    feature.hasFeatureType(“pocket”)) OR
    (product.getInitialProduct.hasShape(“box”) AND
     feature.hasFeatureType(“cylindrical”) AND
     (feature.hasFeatureType(“groove”) OR
      feature.hasFeatureType(“pad”)))
THEN
  RETURN [Capabilities.find(“milling”)]



IF feature.hasCapabilityTaxonomy(“MaterialRemoving”) AND
   feature.hasFeatureType(“thruHole”) AND
   feature.getParam(“thickness”) <= 3.0
THEN
  RETURN [Capabilities.find(“ laserCutting”), Capabilities.find(“punching”)]
```

Taxonomy: MaterialAdding

```
IF feature.hasCapabilityTaxonomy(“MaterialAdding”)
THEN
  RETURN CapabilityTaxonomy.find(“MaterialAdding”).findCapabilities()
```

Taxonomy: Marking

```
IF feature.hasCapabilityTaxonomy(“Marking”)
THEN
  RETURN CapabilityTaxonomy.find(“Marking”).findCapabilities()
```

Detailed capability matching

Taxonomy: Drilling

```
IF feature.getParam(“diameter”) = drillBit.getParam(“hole_diameter”) AND
   feature.getParam(“depth”) <= drillBit.getParam(“max_drilling_depth”) AND
   feature.getParam(“bottomShape”) = drillBit.getParam(“shape_bottom”) AND
   feature.getMaterial() IN  drillBit.suitableMaterials() AND
   product.getInitialProduct().size().isInside(fixturing.minItemsize()-
   fixturing.maxItemSize())
THEN
  RETURN TRUE
```

Taxonomy: Threading

```
IF feature.hasCapabilityTaxonomy(“Threading”) AND
```

```
   feature.getParam(“threadAngle”) = threadingCutter.getParam(“thread_angle”)
   feature.getParam(“threadingStandard”) = threadingCutter.getParam(“thread_standard”)
THEN
  RETURN TRUE
```

Taxonomy: Fixturing

```
fixture = resource.hasCapability(“fixturing”)

IF ((product.getInitialProduct().getParam(“length”) OR
    product.getIntialProduct().getParam(“width”)) <= fixturing.getParam(“max_spread”)) AND
    (product.getIntialProduct().getParam(“length”) OR
     product.getIntialProduct().getParam(“width”)) >= fixturing.getParam(“min_spread”)) AND
    product.getInitialProduct().getParam(“weight”) <=
    fixturing.getParam(“max_item_weight”)) OR
   (product.getInitialProduct().size().isInside(fixture.itemMinSize() –
    fixture.itemMaxSize()) AND
    product.getInitialProduct().getParam(“weight”) <=
    fixture.itemMaxSize.getParam(“weight”))
THEN
  RETURN TRUE
```

Taxonomy: Turning

```
lathe = resource.hasCapability(“spinningWorkpiece”)

IF product.getInitialProduct().size().isInside(lathe.itemMaxSize()) AND
   product.getInitialProduct().getParam(“diameter”) <=
   turningProperties.getParam(“max_turning_diameter”) AND
   feature.getParam(“length”) <= turningProperties.getParam(“max_turning_length”) AND
   product.getMaterial() IN turningTool.suitableMaterials() AND
   feature.hasFeatureType(“type”) = turningTool.getParam(“type”) AND
   feature.getParam(“insideRounding”) >= turningTool.getParam(“nose_radius”) AND
   feature.getParam(“angleTowardsMaterial”) <= (180 – turningTool.getParam(“insert_shape”)
   – turningTool.getParam(“cutting_edge_angle”)) AND
   ((feature.getParam(“angleBetweenFaceAndPad”) <= 90 AND
     feature.getParam(“angleBetweenFaceAndPad”) >=
     turningTool.getParam(“cutting_edge_angle”)) OR
    (feature.getParam(“angleBetweenFaceAndPad”) > 90 AND
     turningTool.getParam(“cutting_edge_angle”) >= (180 – “angleBetweenFaceAndPad”)) AND
  feature.getParam(“tolerance”) >= movingWorkspace.getParam(“repeatability”)
THEN
  RETURN TRUE
```

Taxonomy: Milling

```
millingMachine = resource.hasCapability(“spinningTool”)

IF product.getInitialProduct().size().isInside(millingMachine.itemMaxSize()) AND
   product.getMaterial() IN millingCutter.suitableMaterials() AND
   feature.getParam(“insideRounding”) = millingCutter.getParam(“nose_radius”) AND
   feature.getParam(“tolerance”) >= movingWorkspace.getParam(“repeatability”) AND
   (((feature.hasFeatureType(“pocket”) OR feature.hasFeatureType(“groove”)) AND
      feature.getParam(“width”) >= millingCutter.getParam(“tool_diameter”) OR
    (NOT(feature.hasFeatureType(“pocket”) OR NOT(feature.hasFeatureType(“groove”))) AND
   ((feature.getParam(“outsideRounding”) != 0 AND
     millingCutter.getParam(“type”) = “rounding” AND
     feature.getParam(“outsideRounding”) = millingCutter.getParam(“nose_radius”)) OR
     feature.getParam(“outsideRounding”) = 0) AND
   ((feature.hasFeatureType(“chamfer”) AND
     feature.getParam(“chamferAngle”) = millingCutter.getParam(“cutting_edge_angle”)) OR
    NOT(feature.hasFeatureType(“chamfer”))
THEN
  RETURN TRUE
```

Taxonomy: LaserMarking

```
laserScanner = resource.hasCapability(“laserScanning”)

IF feature.getParam(“length”) <= laserScanner.movingWorkspace.getParam(“length”) AND
   feature.getParam(“width) <= laserScanner.movingWorkspace.getParam(“width”)
THEN
  RETURN TRUE
```

Taxonomy: Screwing

```
IF screw.getParam(“type”) =  screwingHead.getParam(“type”) AND
   screw.getParam(“size”) <= screwingHead.getParam(“screw_size_max”) AND
   screw.getParam(“size”) >= screwingHead.getParam(“screw_size_min”) AND
   screw.getParam(“torque”) <= spinningTool.getParam(“max_torque”) AND
   screw.getParam(“torque”) >= spinningTool.getParam(“max_torque”)
THEN
   RETURN TRUE
```

Taxonomy: VacuumGrasping

```
IF product.getParam(“weight”) <= Combined_payload_of[robot + vacuumGripper] AND
   findResource.hasCapability(“vacuumCup”)
THEN
  RETURN TRUE


IF product.getParam(“weight”) <= Combined_payload_of[robot + vacuumGripper] AND
   findResource.hasCapability(“shapedTip”) AND
   product.size().isInside(shapedTip())
THEN
  RETURN TRUE
```

Taxonomy: MagneticGrasping

```
Magnetic grasping by screwdriver
IF providedCapability = “magneticGrasping” AND
   product.getParam(“material”) = metal AND
   product.getParam(“weight”) <= Combined_payload_of[robot + screwdriver] AND
   product.getParam(“diameter”) <= screwdriver.maxItemSize.getParam(“diameter”) AND
   product.getParam(“length”) <= screwdriver.maxItemSize.getParam(“length”) AND
THEN
   RETURN TRUE
```

Taxonomy: Transporting

```
conveyor = resource.hasCapability(“movingWorkspace”) AND (“holding”)

IF product.getParam(“weight”) <= holding.getParam(“payload”) AND
   product.getParam(“length”) <= conveyor.itemMaxSize.getParam(“length”) AND
   product.getParam(“width”) <= conveyor.itemMaxSize.getParam(“width”) AND
   product.getParam(“height”) <= conveyor.itemMaxSize.getParam(“height)
THEN
   RETURN TRUE
```

NOTE: If other than directly product related parameters are known, e.g. required speed of the transporting capability or required transporting area, the following rules can be applied.

```
With conveyor
IF product.transportingArea(“length”) <= workspaceBox.getParam(“length”) AND
   process.transportingSpeed(“speed”) <= movingWorkspace.getParam(“speed_x”)
THEN
   RETURN TRUE


With robot + gripper
IF process.getParam(“speed_x”) <= movingWorkspace.getParam(“speed_x”) AND
   process.getParam(“speed_y”) <= movingWorkspace.getParam(“speed_y”) AND
   process.getParam(“speed_z”) <= movingWorkspace.getParam(“speed_z”) AND
   process.getWorkspace.size().isInside.Combined_workspace[robot + gripper]
THEN
   RETURN TRUE
```

Taxonomy: PlateFeeding

```
plate = resource.hasCapability(“feedingPlate”)

IF product.getParam(“diameter”) <= plate.itemMaxSize.getParam(“diameter”) AND
   product.getParam(“length”) <= plate.itemMaxSize.getParam(“length”) AND
THEN
   RETURN TRUE
```

Taxonomy: ImageCapturing

```
IF product.getSmallestParam.size() >= 2 * FoV width /
   imageCapturing.getParam(“x_resolution”) AND
   product.getSmallesParam.size() >= 2 * FoV height /
   imageCapturing.getParam(“y_resolution”)
THEN
   RETURN TRUE
```

COMBINED CAPABILITY RULES

Combining parameters

Tool length (tool + tool holder)

Taxonomy: Milling

```
millingCutter = resource.hasCapability(“millingCutter”)

Max_length_of_the _tool =
toolHolder.getParam(“effective_length”) + millingCutter.BasicDeviceInfo.getParam(“length”)
– MAX(toolHolder.getParam(“tool_entry_min”), millingCutter.getParam(“min_entry”))

Min_length_of_the_tool =
toolHolder.getParam(“effective_length”) + millingCutter.BasicDeviceInfo.getParam(“length”)
– MIN(toolHolder.getParam(“tool_entry_max”), millingCutter.getParam(“max_entry”))

Current_length_of_the_tool =
toolHolder.getParam(“effective_length”) + millingCutter.getParam(“current_entry”)
```

Taxonomy: Drilling

```
drillBit = resource.hasCapability(“drillBit”)

Max_length_of_the _tool =
toolHolder.getParam(“effective_length”) + drillBit.BasicDeviceInfo.getParam(“length”) –
MAX(toolHolder.getParam(“tool_entry_min”), drillBit.getParam(“min_entry”))

Min_length_of_the_tool =
toolHolder.getParam(“effective_length”) + drillBit.BasicDeviceInfo.getParam(“length”) –
MIN( toolHolder.getParam(“tool_entry_max”), drillBit.getParam(“max_entry”))

Current_length_of_the_tool =
toolHolder.getParam(“effective_length”) + drillBit.getParam(“current_entry”)
```

Workspace

Taxonomy: Machining

```
Workspace in horizontal machining centers

Vertical reach of the tool
Lowest_spot_in_the_initial_product_where_can_be_machined =
millingProperties.getParam(“distance_spindle_table_top_min”) –
fixturing.getParam(fixturing_position_z)
Highest_spot_in_the_initial_product_where_can_be_machined =
millingProperties.getParam(“distance_spindle_table_top_max”) –
fixturing.getParam(“fixturing_position_z”)

Horizontal reach of the tool
Minimum_distance_between_pallet_center_and_tool =
millingProperties.getParam(“distance_spindle_pallet_center_min”) – Max_length_of_the_tool
Maximun_distance_between_pallet_center_and_tool =
millingProperties.getParam(“distance_spindle_pallet_center_max”) – Min_length_of_the_tool
Current_maximum_distance_between_pallet_center_and_tool =
millingProperties.getParam(“distance_spindle_pallet_center_max”) –
Current_length_of_the_tool
```

Taxonomy: ObjectRecognition

```
FoV_width = (lightReflecting.getParam(“current_working_distance”) *
imageCapturing.getParam(“detector_size_x”)) / lightReflecting.getParam(“focal_length_min”)

FoV_height = (lightReflecting.getParam(“current_working_distance”) *
imageCapturing.getParam(“detector_size_y”)) / lightReflecting.getParam(“focal_length_min”)
```

Payload

Taxonomy: Machining

```
Payload of machining center
millingMachine = resource.hasCapability(“spinningTool”)
fixture = resource.hasCapability(“fixturing”)

Combined_payload_of[millingMachine + fixture] =
MIN(millingMachine.payload.getParam(“weight”) - fixture.basicDeviceInfo.getParam(“weight”),
fixturing.getParam(“max_item_weight”))
```

Taxonomy: Manipulation

```
Payload of robot + finger gripper
robot = resource.hasCapability(“movingWorkspace”)
gripper = resource.hasCapability(“fingerGripper”)
finger = resource.hasCapability(“finger”)

Combined_payload_of[robot + fingerGripper + fingers] = MIN(robot.payload.getParam(“weight”)
– gripper.basicDeviceInfo.getParam(“weight”) – fingerGripper.getParam(“number_of_fingers”)
* finger.basicDeviceInfo.getParam(“weight”),
fingerGripper.getParam(“payload”))

Payload of robot + vacuum gripper
```

```
robot = resource.hasCapability("movingWorkspace")
vacuumCreator = resource.hasCapability("vacuumCreation")

Combined_payload_of[robot + vacuumGripper] = MIN(robot.payload.getParam("weight") –
vacuumCreator.basicDeviceInfo.getParam("weight") -vacuumCreation.getParam("number_of_cups")
* vacuumCup.getParam("weight")),
1/9,81 * vacuumCup.getParam("holding_force") * vacuumCreation.getParam("number_of_cups"))

Payload of robot + screwdriver
robot = resource.hasCapability("movingWorkspace")
screwdriver = resource.hasCapability("spinningTool")
screwinghead = resource.hasCapability("screwingHead")

Combined_payload_of[robot + screwdriver] = MIN(robot.payload.getParam("weight") -
screwdriver.basicDeviceInfo.getParam("weight") –
screwinghead.basicDeviceInfo.getParam("weight")),
1/9,81 * magneticGrasping.getParam("holding_force"))
```

Taxonomy: TrayFeeding, PlateFeeding

```
Payload of the plate feeder
trayFeeder = resource.hasCapability("traySupporting")
plate = resource.hasCapability("feedingPlate")

Combined_payload_of[feeder + plate] = traySupporting.getParam("payload) –
(traySupporting.getParam("number_of_trays") * plate.basicDeviceInfo.getParam("weight"))
```

Accuracy

Taxonomy: Manipulation

```
Placing and inserting accuracy of robot and 2-finger gripper
Combined_accuracy[robot + 2-finger gripper] = movingWorkspace.getParam("accuracy") +
2*finger.getParam("tolerance") + 2*fingerGripper.getParam("accuracy")
```

Taxonomy: Machining

```
Machining accuracy [machine + fixture]
machine = resource.hasCapability("spinningWorkpiece") OR
resource.hasCapability("spinningTool")

Machining_accuracy[machine + fixture] = movingWorkspace.getParam("accuracy") +
fixturing.getParam("tolerance")
```

Combining interfaces

Tool + Tool holder

Taxonomy: Milling

```
IF toolHolding.getParam("tool_attach_diameter_type") !=
   millingCutter.getParam("holder_attach_diameter_type") AND
   toolHolding.getParam("tool_attach_diameter_min") <=
   millingCutter.getParam("holder_attach_diameter") AND
   toolHolding.getParam("tool_attach_diameter_max") >=
   millingCutter.getParam("holder_attach_diameter")
THEN
  RETURN TRUE
```

Taxonomy: Drilling

```
IF toolHolding.getParam("tool_attach_diameter_type") !=
   drillBit.getParam("holder_attach_diameter_type") AND
   toolHolding.getParam("tool_attach_diameter_min") <=
   drillBit.getParam("holder_attach_diameter") AND
   toolHolding.getParam("tool_attach_diameter_max") >=
   drillBit.getParam("holder_attach_diameter")
THEN
  RETURN TRUE
```

Taxonomy: Threading

```
IF toolHolding.getParam("tool_attach_diameter_type") !=
   threadingCutter.getParam("holder_attach_diameter_type") AND
   toolHolding.getParam("tool_attach_diameter_min") <=
   threadingCutter.getParam("holder_attach_diameter") AND
   toolHolding.getParam("tool_attach_diameter_max") >=
   threadingCutter.getParam("holder_attach_diameter")
THEN
  RETURN TRUE
```

Taxonomy: Screwing

```
screwdriver = resource.hasCapability("spinningTool")
screwinghead = resource.hasCapability("screwingHead")

IF (spinningTool.getParam("diameter_max") >=
   screwingHead.basicDeviceInfo.getParam("diameter")) OR
   (screwdriver.OutputInterface() = screwingHead.InputInterface())
THEN
  RETURN TRUE
```

Machine + tool holder

Taxonomy: Machining

```
machine = resource.hasCapability("spinningWorkpiece") OR
resource.hasCapability("spinningTool")
toolHolder = resource.hasCapability("toolHolding")

IF machine.hasOutputInterface("X") = toolHolder.hasInputInterface("X")
THEN
  RETURN TRUE
```

ADAPTATION RULES

Generic guidelines

```
IF requiredCapability.ExistsIn.currentSystem.findCapabilities() AND
   matchParameters(currentSystem.capability.find(capabilityParameters),
   requiredCapability(requiredParameters)) = true
THEN
   Use the current system and apply needed logical and parametric adaptation actions
ELSE
   Perform physical adaptation actions (reconfiguration)


IF Physical adaptation of the existing system is needed
THEN
  Minimize the adaptation distance and create new combinations of the existing resources
  and resources in storage
```

New combination generation

Taxonomy: Turning

```
IF feature.hasCapabilityTaxonomy ("Turning") AND currentSystem.hasCapability("turning") AND
   matchParameters(currentSystem.capability.find(capabilityParameters),
   requiredCapability(requiredParameters)) = false
THEN
   Create new combinations of the resource.hasCapability("spinningWorkpiece") and other
   resource.hasCapability("turningTool")
```

Taxonomy: Screwing

```
IF feature.hasCapabilityTaxonomy ("Screwing") AND
   currentSystem.hasCapability("screwing") AND
   matchParameters(currentSystem.capability.find(capabilityParameters),
   requiredCapability(requiredParameters)) = false
THEN
   Create new combinations of the resource.hasCapability("spinningTool") and other
   resource.hasCapability("screwingHead")
```

Taxonomy: TrayFeeding, PlateFeeding

```
IF process.hasCapabilityTaxonomy ("PlateFeeding") AND
   currentSystem.hasCapability("plateFeeding") AND
   matchParameters(currentSystem.capability.find(capabilityParameters),
   requiredCapability(requiredParameters)) = false
THEN
   Create new combinations of the resource.hasCapability("traySupporting") and other
   resource.hasCapability("feedingPlate")
```

Context specific rules

User defined rules

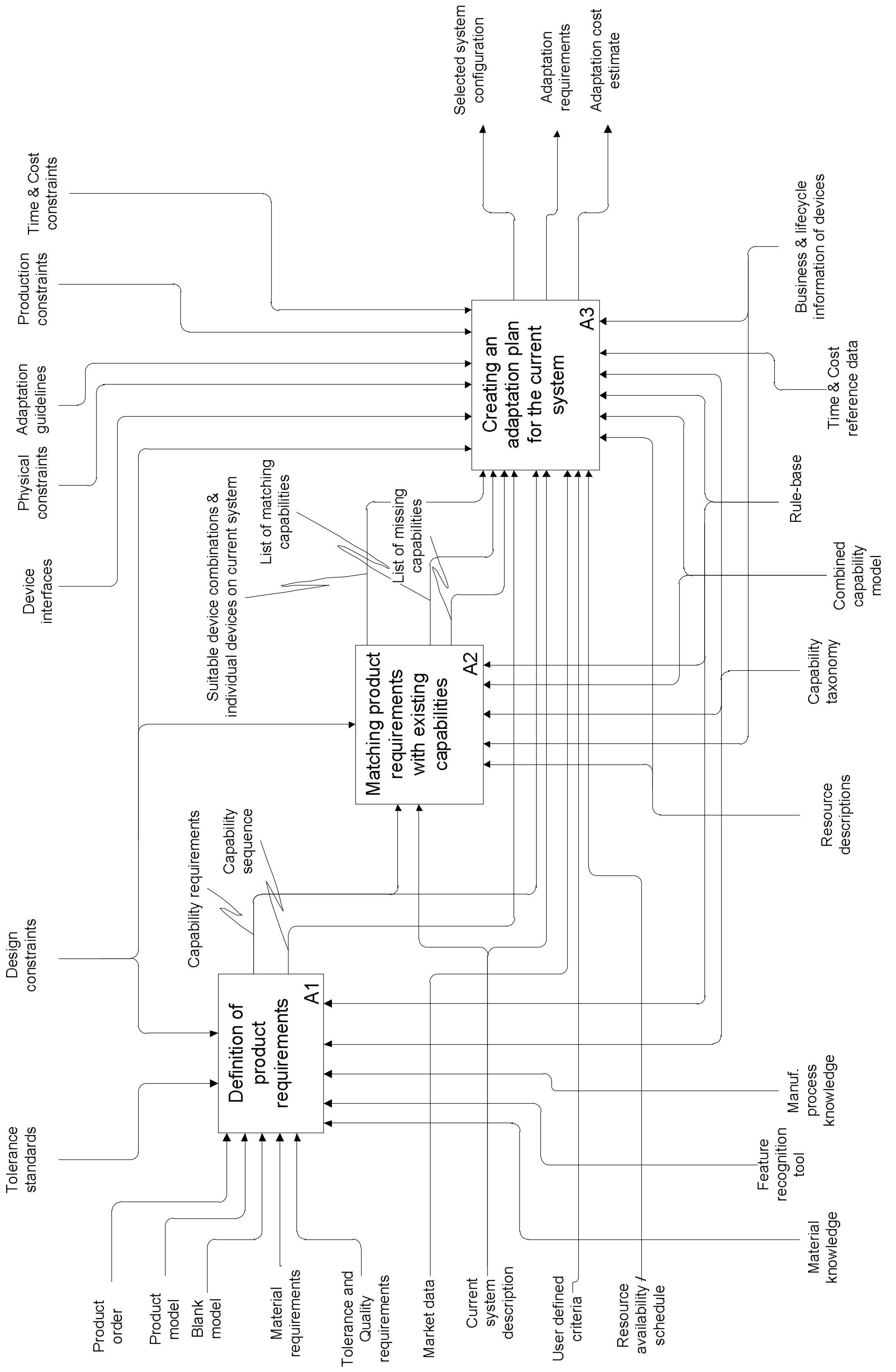
```
IF order.getParam("number_of_ordered_items") >= User given value
THEN
   RETURN resource.hasProperty("fastest")
```

```
IF order.getParam(“number_of_ordered_items”) < User given value
THEN
  RETURN resource.hasProperty(“cheapest”)
```

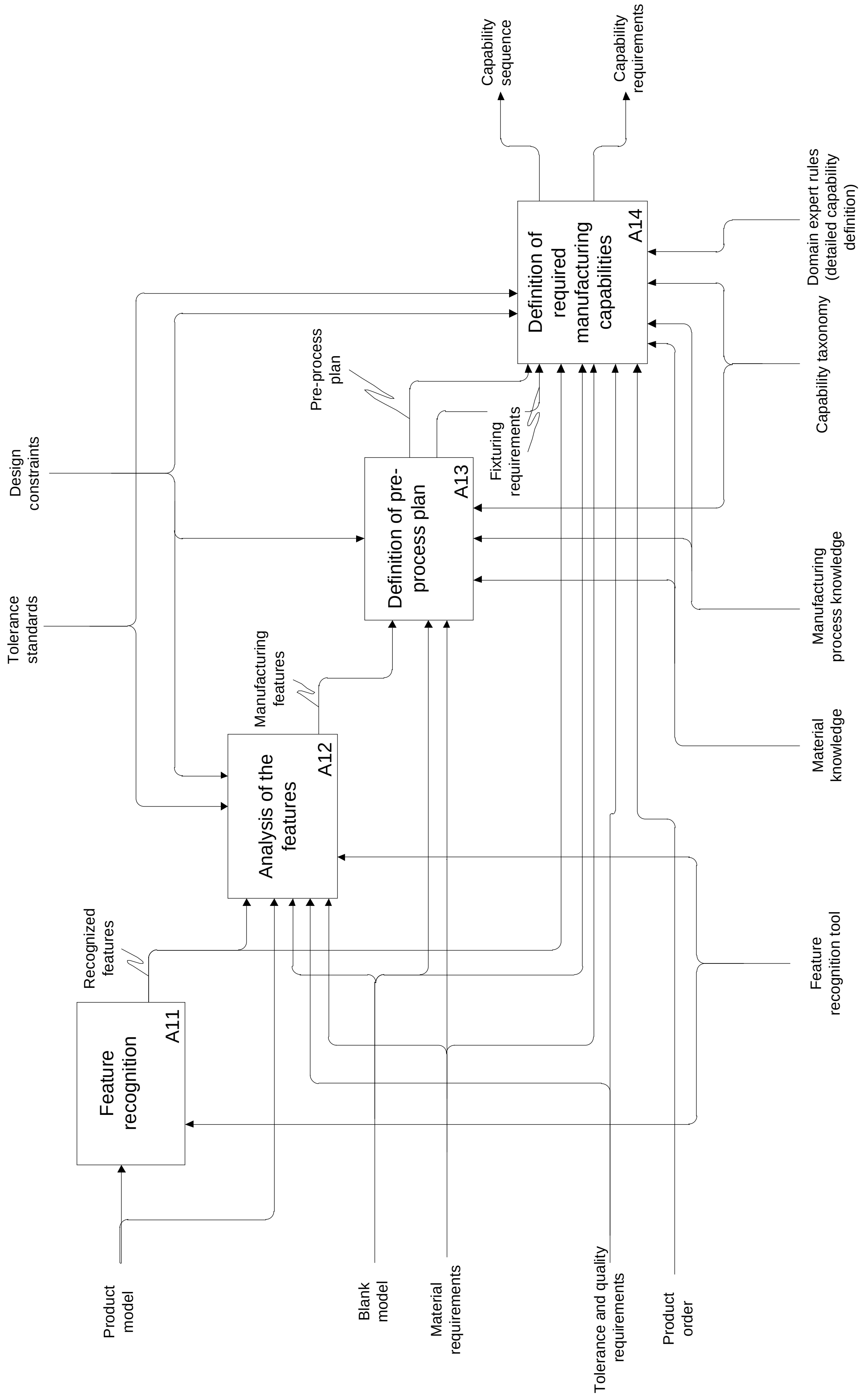
Rules for applying user given criteria

```
IF Reliability of the resource is specially valued
THEN
  RETURN MAX(resource.hasLifecycleProperty(“MTBF”))


IF Environmental sustainability is specially valued
THEN
  RETURN MIN(resource.hasProperty(“energy_consumption”))
```

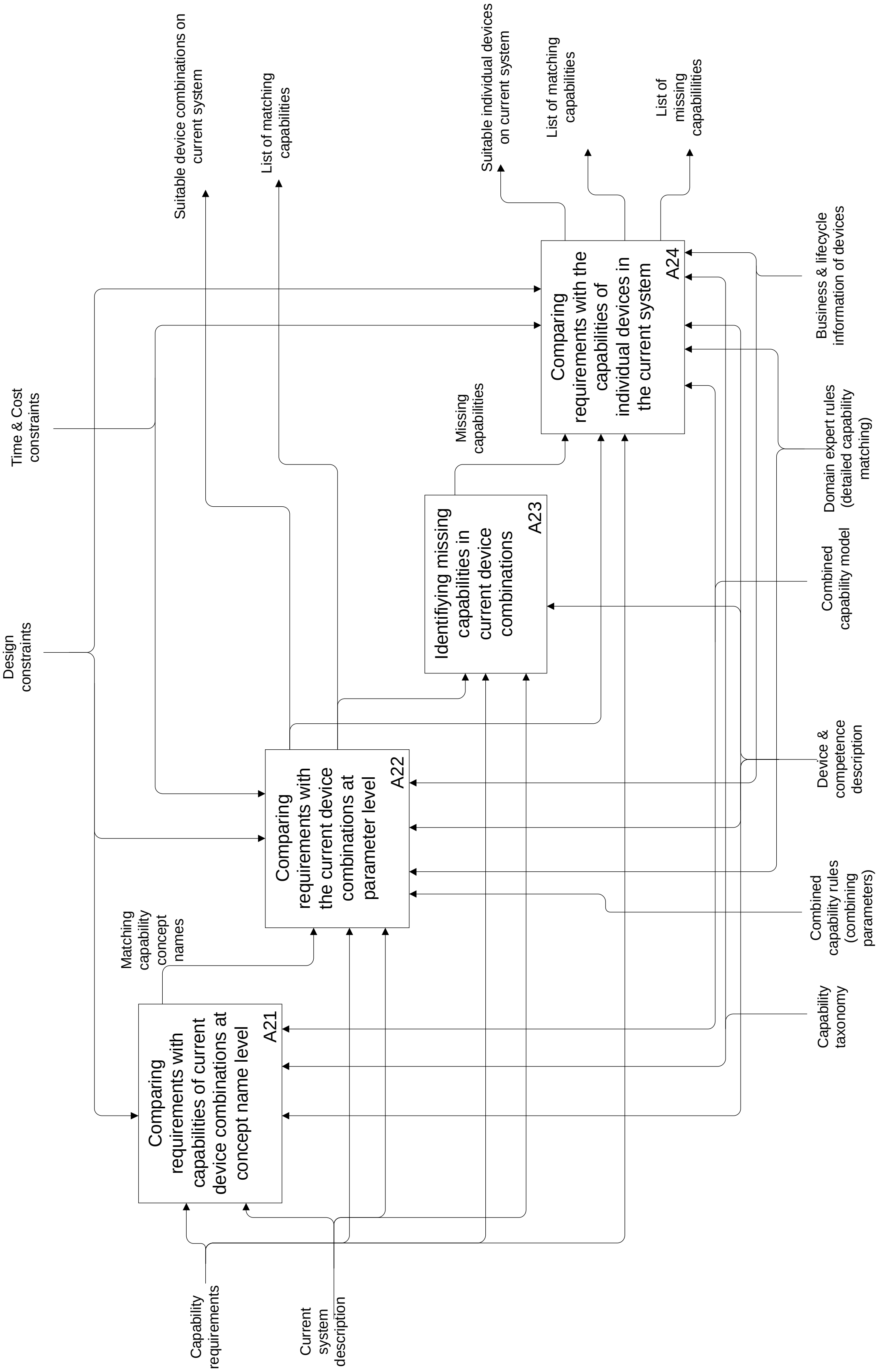
Appendix 6: Adaptation schema – Node A0, Overall adaptation schema


Definition of product requirements
A1
Matching product requirements with existing capabilities
A2
Creating an adaptation plan for the current system
A3
Tolerance standards
Design constraints
Device interfaces
Physical constraints
Adaptation guidelines
Production constraints
Time & Cost constraints
Selected system configuration
Adaptation requirements
Adaptation cost estimate
Business & lifecycle information of devices
Time & Cost reference data
Rule-base
Combined capability model
Capability taxonomy
Resource descriptions
Manuf. process knowledge
Feature recognition tool
Material knowledge
Product order
Product model
Blank model
Material requirements
Tolerance and Quality requirements
Market data
Current system description
User defined criteria
Resource availability / schedule
Capability requirements
Capability sequence
Suitable device combinations & individual devices on current system
List of matching capabilities
List of missing capabilities

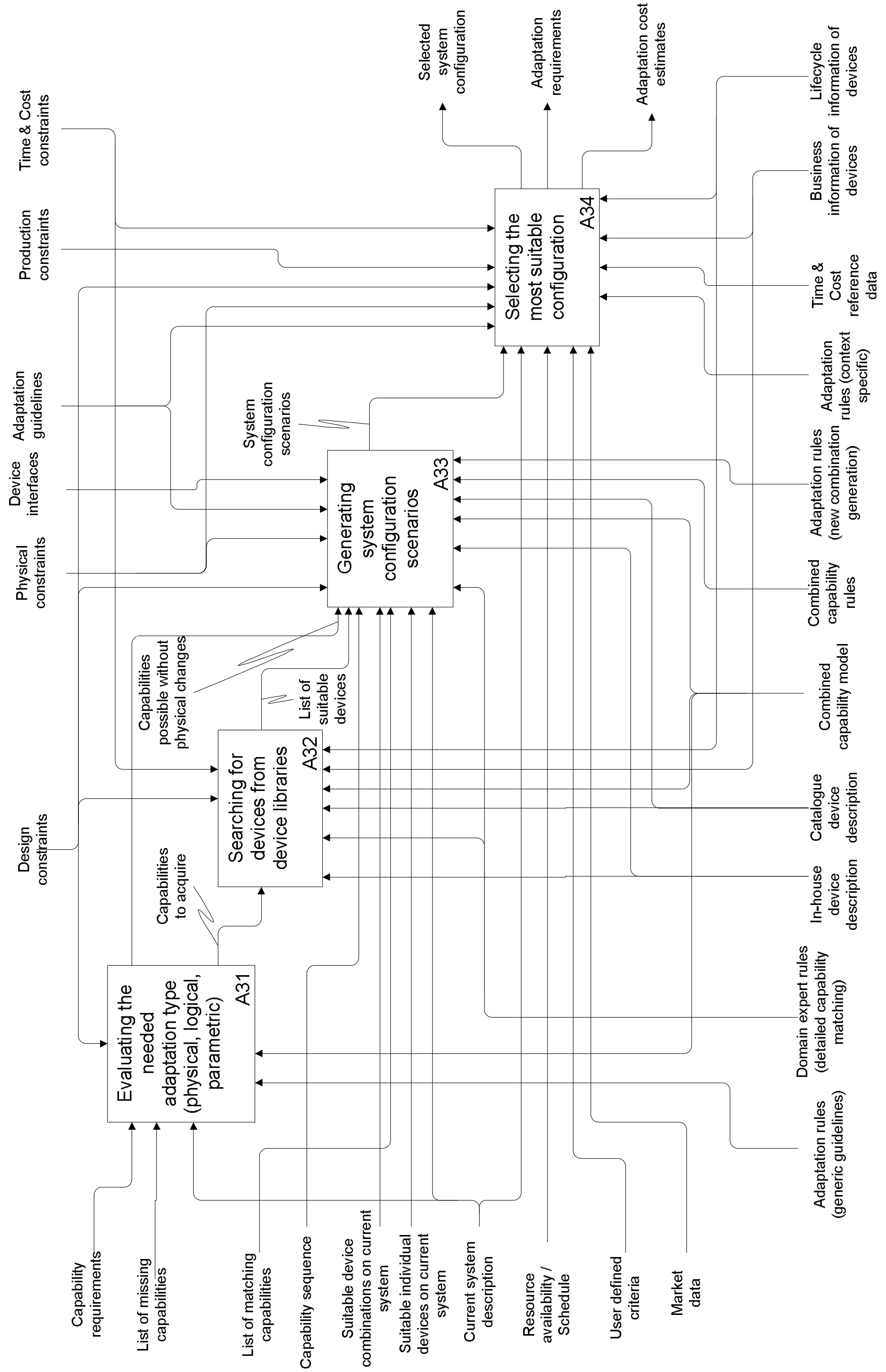
Appendix 7: Adaptation schema – Node A1, Definition of product requirements



Appendix 8: Adaptation schema – Node A2, Matching product requirements with existing capabilities


Comparing requirements with capabilities of current device combinations at concept name level
A21
Comparing requirements with the current device combinations at parameter level
A22
Identifying missing capabilities in current device combinations
A23
Comparing requirements with the capabilities of individual devices in the current system
A24
Time & Cost constraints
Design constraints
Matching capability concept names
Missing capabilities
Suitable device combinations on current system
List of matching capabilities
Suitable individual devices on current system
List of matching capabilities
List of missing capabilities
Business & lifecycle information of devices
Domain expert rules (detailed capability matching)
Combined capability model
Device & competence description
Combined capability rules (combining parameters)
Capability taxonomy
Capability requirements
Current system description

Appendix 9: Adaptation schema – Node A3, Creating an adaptation plan for the current system


Time & Cost constraints
Production constraints
Adaptation guidelines
Device interfaces
Physical constraints
Design constraints
Selected system configuration
Adaptation requirements
Adaptation cost estimates
Lifecycle information of devices
Business information of devices
Time & Cost reference data
Adaptation rules (context specific)
Adaptation rules (new combination generation)
Combined capability rules
Combined capability model
Catalogue device description
In-house device description
Domain expert rules (detailed capability matching)
Adaptation rules (generic guidelines)
Selecting the most suitable configuration
A34
System configuration scenarios
Generating system configuration scenarios
A33
Capabilities possible without physical changes
List of suitable devices
Searching for devices from device libraries
A32
Capabilities to acquire
Evaluating the needed adaptation type (physical, logical, parametric)
A31
Capability requirements
List of missing capabilities
List of matching capabilities
Capability sequence
Suitable device combinations on current system
Suitable individual devices on current system
Current system description
Resource availability / Schedule
User defined criteria
Market data

Appendix 10: Adaptation schema – Description of the terms used in the IDEF0-diagrams

Table 35. Description of the inputs, outputs, controls and resources used in the adaptation schema IDEF0-diagrams in alphabetical order.

TERM	EXPLANATION
Adaptation cost estimate	Estimation of the total cost of adapting the production system, including layout, devices and tooling, installation, programming, training, etc.
Adaptation requirements	Adaptation requirements define how the current system needs to be modified in order to meet the new requirements. These changes include the needed physical changes, such as adding devices, logical changes, such as changing the product routing, or parametric changes, such as changing the processing parameters of the machines.
Adaptation guidelines	Context and domain specific guidelines that guide the adaptation planning process. The guidelines are defined by the company policies, philosophy and way of doing things. They may guide, for example, to what extent devices should be re-used and how the trade-off between new, efficient but expensive equipment and old, cheap but slow equipment should be handled.
Adaptation rules	Adaptation rules are divided into three categories: generic guidelines, new combination generation, and context-specific rules. The generic guidelines provide rules for determining which type of adaptation is required, namely physical, logical or parametric. The new combination generation rules guide the generation of new combinations from existing and acquired devices. The context specific rules define how the context or user specific criteria is applied when making the selection between different configuration scenarios.
Blank model	The blank model describes the original shape and dimensions of the part (billet) to be processed.
Business information of devices	Business information includes information relating to the costs of acquiring and using the specific devices.
Capabilities possible without physical changes	Capabilities which can be obtained without making any physical changes to the current system. This means that only logical or parametric changes need to be made.
Capability requirements	Capability requirements are functional requirements for resources and are set by the product characteristics. These are the specific capabilities required for the production of a particular product with the required quality and volume. Capability requirements include the requirement for the functional capability (e.g. “moving”, “grasping”, “drilling”) as well as the capability parameter requirements (e.g. “velocity” of the “moving” capability). Capability requirements can include both simple and combined capabilities and can be defined at different taxonomy levels. This means that the functional requirement can be defined as e.g. “material removing”, or with more detail e.g. “milling” or “turning”.
Capabilities to acquire	Indicates the capabilities which do not exist in the current production system and can not be obtained by making logical or parametric adaptations to the system. Therefore, these capabilities needs to be acquired from outside the current system. The missing capabilities may be found from the company’s own device pool, or otherwise they may need to be imported from outside.
Capability sequence	The order in which the capabilities should take place during the production process in order to manufacture or assemble the product. For example, in the case of assembly, it means the sequence in which the assembly operation-related capabilities should be performed.
Capability taxonomy	A taxonomy of existing and possible capabilities. The taxonomy allows the linking of capabilities at different abstraction levels and the search for devices which provide similar

	functions (e.g. joining) with different behaviors (e.g. welding, gluing, riveting). Both the product requirements and device capabilities refer to this taxonomy, enabling the mapping of the capabilities at different abstraction levels.
Catalogue device descriptions	Information about devices that can be ordered from the equipment and system providers.
Combined capability model	A model, and its instances, indicating which simple capabilities are needed in order to achieve combined capabilities.
Combined capability rules	Combined capability rules are divided into two categories: Combining parameters and Combining interfaces rules. The first ones are used to combine the parameters of simple capabilities in order to derive the parameters of the combined capabilities, which consist of these simple capabilities. The latter rules are used to check that the devices have compatible interfaces, when creating new device combinations.
Competence description	A description of the competences possessed by the human resources of the factory. This includes both standard competences and special competences and includes the ability to perform specific fabrication, assembly, and inspection operations and/or the ability to achieve certain levels of quality in performing the operations.
Current system description	A description of the existing production system including both the machine resources (devices) and human resources. It also describes the layout of the current system.
Design constraints	Constraints on the production system designs imposed by corporate policy, layout, system reference architecture, technology used, ergonomics, or the paradigm which is followed.
Device description	A technical description of the production device resources which may be available for manufacturing a product. The devices include, for example, fabrication machines (turning, milling, punching,…); assembly machines (robots, conveyors, grippers, …); inspection machines; material handling and transport devices; tools (milling and turning cutters, drill bits, holders…); and fixtures. The resource description of the devices covers the following important aspects: the capability of the device, including the capability concept name and parameters, interfaces and business and lifecycle information.
Device interfaces	A description of the device interfaces, including mechanical, energy and communication interfaces. They allow the matching of devices with other devices having compatible interfaces.
Domain expert rules	Domain expert rules are divided into two categories: detailed capability definition; and, detailed capability matching rules. The first ones are used to define the detailed capability requirements from the pre-process plan, whereas the latter ones enable a detailed match between the required and provided capabilities to be made.
Feature recognition tool	The feature recognition tool is able to recognize and classify features from the 3D product model. With the tool, it is possible to analyze what kind of manufacturing and assembly processes are required during the production of the product.
Fixturing requirements	The fixturing requirements specify the requirements concerning the billet set-up and fixturing.
In-house device descriptions	A resource description of the devices that exist in the factory, either on the current system layout or in storage.
Lifecycle information of devices	This includes information about the usage, maintenance history and condition of the devices. It also includes data collected from the factory floor, e.g. MTBF and operational values.
List of matching capabilities	A list of all the capability requirements that can be satisfied, including descriptions of the devices that provide those capabilities.
List of missing capabilities	A list of capabilities that are missing from the current system and need to be obtained by either physical, logical or parametric adaptation actions.
List of suitable devices	A list of devices whose capabilities match the current requirements.

Manufacturing features	The principal features of the product or part, which affect processing decisions.
Manufacturing process knowledge	General knowledge of the manufacturing processes which could be used to make a given component or subsystem, and the limitations imposed by such processes.
Market data	Estimates of the total volume of the product which could be sold, its market life, and the production rate required to meet the projected market opportunity. In a dynamic market environment there may not be reliable estimates available.
Matching capability concept names	A list of capabilities matching the requirements at the capability concept name level. The concept name level match is found with the help of the capability taxonomy.
Material knowledge	Information about the characteristics of raw materials, such as structural, chemical, thermal and electrical properties. This includes engineering knowledge about the material behavior with different processes and process parameters. It helps to define suitable manufacturing and assembly methods for the product.
Material requirements	Special requirements concerning the production processes and process parameters set by the specific material characteristics.
Missing capabilities	Those required capabilities which do not have a match in device combinations on the current system (e.g. on current workstations). This indicates that the combined capabilities need to be divided into simple capabilities and a match should be searched for from the individual devices' capabilities.
Physical constraints	The dimensional constraints set by the facility space and other devices on the factory floor.
Pre-process plan	A preliminary definition of the manufacturing and assembly process based on the identified manufacturing features. The pre-process plan is a generic recipe of how to produce the product and it only defines the required functions on high level, e.g. "joining" or "material removing".
Production constraints	The requirements and limitations on facility adaptation which are derived from ongoing production management considerations, including current and future production commitments.
Product model	This is a computer-interpretable representation of the product and its assembly structure. It typically includes a 3D CAD-model of the product and its components, their geometry, dimensions, materials, tolerances and surface finishes.
Product order	The quantities of parts or products to be produced and their nominal delivery dates.
Recognized features	The principal features affecting processing decisions, identified from the product model. These include, for example, the shape, type and dimensions of the feature.
Resource availability	The resource availability indicates the current and planned levels of utilization of the resources in the production facility. This shows whether the resource is currently allocated to another production task or if it is freely available. It also includes information about the resource state. Such information could include: busy on other job, idle, broken, down for maintenance, etc.
Resource descriptions	Technical descriptions of the production resources which may be available for the production of the product. The resources include both machine (device) and human resources. In other graphs the resource descriptions are further divided into device description, competence description, current system description, in-house device description and catalogue device description.
Resource schedule	The expected schedule of availability of resources for production tasks.
Rule-base	The rule-base contains the Domain expert rules, Combined capability rules and Adaptation rules that are used for the automatic matching of product requirements and system

	capabilities during the adaptation planning.
Selected system configuration	The selected system configuration is the system configuration that is seen as feasible for the given production problem in the specific context. The selection is based on the capabilities and availability of the system and different user-defined criteria, such as costs, speed or energy efficiency as well as different company and domain-specific constraints and the condition of the resources.
Suitable individual devices on current system	Those individual devices on the current system that possess capabilities matching the capability requirements.
Suitable device combinations on current system	The combinations of devices on the current system that possess capabilities matching the capability requirements. These combinations can be, for example, complete workstations or cells.
System configuration scenarios	Different system configuration scenarios matching the capability requirements set by the product order.
Time & Cost constraints	The limitations imposed by product planning, and other corporate decisions regarding the acceptable manufacturing costs and time-to-market for a product.
Time & cost reference data	Standardized data that specify estimated times and costs for standard operations and can be used in the preliminary cost estimates.
Tolerance and quality requirements	The requirements for part tolerances and quality.
Tolerance standards	These are the industrial or corporate standard engineering tolerances, tolerancing techniques and limitations.
User defined criteria	The various user-defined criteria that are used to compare different production system configuration scenarios against each other. These criteria may relate, for example, to costs, speed or energy efficiency.

Tampereen teknillinen yliopisto
PL 527
33101 Tampere

Tampere University of Technology
P.O.B. 527
FI-33101 Tampere, Finland

ISBN 978-952-15-2928-3
ISSN 1459-2045