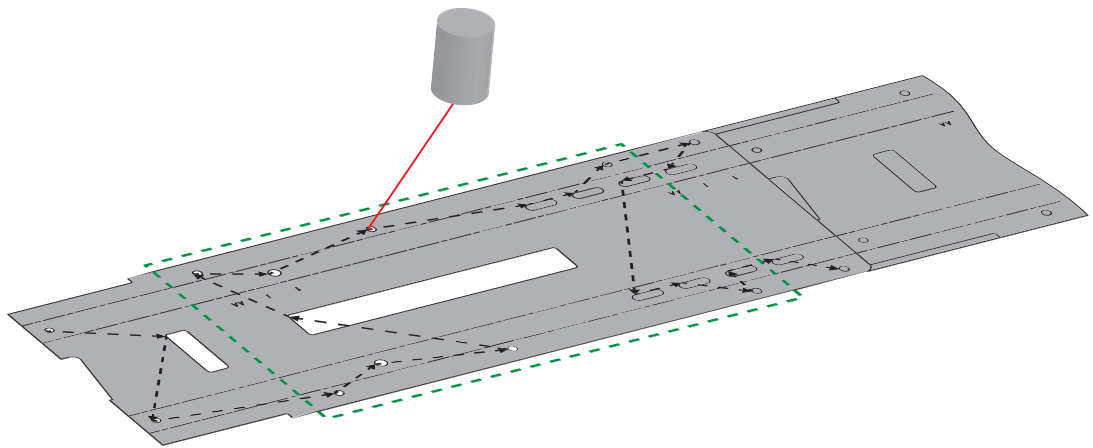


ROBOCUT

Scheduling and Configurator



MASTER'S THESIS

JONATHAN SKOVHUS ANDERSEN & DAN GADENSGAARD

Department of Mechanical and Manufacturing Engineering

Aalborg University, Denmark

Manufacturing Technology

Fibigerstræde 16,
DK-9220 Aalborg Ø
Tlf: +45 9940 7117
Fax: +45 9940 7110
<http://www.m-tech.aau.dk>

Title:

ROBOCUT

Semester Theme:

Master's Thesis

Project Period:

01.02.2011 - 07.06.2011

Authors:

Jonathan Skovhus Andersen

Dan Gadensgaard

Supervisors:

Morten Kristiansen
Ewa Kolakowska

Publications: 4.

Number of Pages: 138 + appendix
(18 blank pages)

Finished: 07.06.2011

Synopsis:

This thesis is a part of the ROBOCUT research project. The project's vision is to develop a new laser cutting technology, that relies on a multibeam principle in order cut remotely.

This thesis is split into two parts. Part I focuses on scheduling of the laser cutting process. A cutting job is divided into a set of cutting tasks. Methods for scheduling these tasks in order to minimize process time is evaluated. Possible methods are: Use of dispatching rules or combinatorial optimization. The dispatching rules can be used to construct a path that is used as input for an improvement heuristic. Using combinatorial optimization is only possible for up to around 25 tasks because of computation time.

Part II defines a methodology for modularizing a production unit and preparing it for configuration. The output of this methodology is used to develop an online configuration system using ASP.NET C#.

Preface

This thesis is written by Jonathan Skovhus Andersen and Dan Gadensgaard during the spring semester 2011. The work was carried out as a part of the 10th semester of the *Manufacturing Engineering and Technology* candidate programme at Aalborg University, Aalborg, Denmark. The overall theme of the project is "*Technologically innovative business creation*", and the overall purpose of the project follows the curriculum of the candidate programme in which it is stated that the students are expected to:

- ◇ Be able to acquire new knowledge required to solve an industrial or scientific problem within manufacturing engineering and technology.
- ◇ Be able to demonstrate engineering and/or scientific skills within the line of specialization and to display their ability to perform engineering and/or scientific work.
- ◇ Be able to work independently with a project on a specific problem within their field of interest on the highest possible level within their specialization.

The project is documented by a main report, appendix, list of references and an enclosed CD. The main report consists of two main parts that can be read independently, as well as a conclusion.

References are presented according to the Chicago Method, which in the text are indicated as [x, year] where "x" is the author's surname and "year" is the year of publication. If multiple references are used, these will be printed [x, year], [y, year]. References with specific page numbers are indicated as [x, year, p. a-b]. If referring to an author who published several articles in the same year, the first material is declared with an "a" after "year". The other publications of the same author are then given b, c, d... depending on the order of its appearance in the report.

The reference list generally specifies as follows:

Surname, first name. Year. Title. Publishing / URL.

Graphs, formulas and tables are numbered sequentially, with chapter number and then figure number, e.g. "Figure 4.1". In the appendices letters indicate chapters.

On the last page a CD is enclosed. This is also divided into two main parts, containing:

- ◇ Literature
- ◇ The report in PDF.
- ◇ Part I
 - MATLAB scripts for nearest neighbor and dispatching rule schedulers.
 - Gecode source code and executables for combinatorial optimization.
 - Visual C# project and source code for ROBOCUTAddIn.
- ◇ Part II
 - PVM model and Product Model Manager software.
 - ASP.NET and Visual C# source code for ROBOCUT Configurator.
- ◇ Pictures from visit at Ib Andresen A/S.

Contents

1	Introduction	1
1.1	The ROBOCUT Project	2
1.2	Scope of this Project	6
2	Problem Formulation	9
I	Scheduling System	11
3	Introduction	13
3.1	Integrated Planning and Scheduling	14
3.2	Introduction to Scheduling	16
3.3	Scheduling Definitions and Notations	17
3.4	Related Work	18
4	The Scheduling Problem	21
4.1	Usage Scenarios	21
4.2	Scheduling the RLC Process for Roll Forming	23
4.3	Mathematical Description of the Problem	24
4.4	Assumptions and Approach	25
5	Dispatching Rule Scheduler	29
5.1	Basic Dispatching Rules	30
5.2	Composite Dispatching Rules	35
5.3	Summary	41

6	Combinatorial Optimization	43
6.1	Running Time of Algorithms	43
6.2	The Travelling Salesman Problem	44
6.3	Software	51
6.4	Implementation	53
6.5	Results	55
6.6	Summary	63
7	Scheduling Interface	65
7.1	Framework	65
7.2	Requirements	67
7.3	The Robocut Add-In	68
7.4	Summary	71
8	Discussion	73
II	ROBOCUT Configurator	77
9	Introduction	79
9.1	Notation	80
10	Identifying Modular Architecture	83
10.1	Clarifying the Task	85
10.2	Establishing Function Structures	91
10.3	Working Principles and Variants	95
10.4	Selecting and Evaluating modules	100
10.5	Embody Modular Architecture and Framework	101
11	Software Requirements	111
11.1	Scope	112
11.2	Product Perspective	112

11.3	Specific Requirements	113
11.4	Summary	115
12	Software Design Specification	117
12.1	Architecture	117
12.2	Database Design	120
12.3	User Interface Design	124
13	Verification and Validation	127
13.1	Presentation of Configurator	127
13.2	Validation Testing	128
13.3	Defect Testing	131
13.4	Summary	132
14	Discussion	133
III	Conclusion	135
15	Conclusion	137
	Summary	138
	Bibliography	141
IV	Appendix	149
A	Ib Andresen Industri A/S	A-1
A.1	Roll forming	A-2
A.2	ROBOCUT potential in the roll forming process	A-4
A.3	Visits at Ib Andresen	A-6
A.4	Problems associated with punched holes	A-7
A.5	CAD data	A-7

A.6	Changeover	A-8
A.7	Software for defining holes	A-8
A.8	Controlling the laser cutting process	A-8
A.9	Setup of Laser	A-8
B	Scheduling Constraints	B-1
B.1	Description of Constraints	B-1
C	Gecode Source Code	C-1
D	Functional Structure Diagrams	D-1
D.1	RLC ahead of or inside the roll forming mill at IAI	D-2
D.2	RLC after the roll forming mill at IAI	D-3
D.3	Robotic RLC at Grundfos	D-4
D.4	Station RLC at Grundfos	D-5
D.5	Robotics RLC of holes in car bodies at Volvo	D-6
E	Requirements	E-1
E.1	Functional Requirements	E-1
E.2	Non-Functional Requirements	E-2
F	User Manuals	F-1
F.1	Installing and running the ROBOCUT Add-in	F-1
F.2	Using the ROBOCUT Add-in	F-2
F.3	User Manual for ROBOCUT Configurator	F-7

CHAPTER 1

Introduction

As laser was discovered in 1960 it was said to be "a solution looking for a problem" [Steen and Mazumder, 2010, p. 51]. The tool was so different from anything that had ever been seen that the current thinking had not caught up with the possibilities. Today a laser is associated with precision, quality and speed. Desktop laser printers, DVD's, a laser scanner at the supermarket are examples of everyday appliances where the use of lasers is taken for granted.

The history of laser technology actually starts way back in 1917 when Albert Einstein established the theoretic foundations for the laser¹ and the maser². Albert Einstein showed that stimulated emission - the basis for generating laser radiation - is an everyday occurrence. His discoveries lead to considerable investments in research and in 1960 Theodore Harold "Ted" Maiman (1927-2007) invented the first working ruby laser [Steen and Mazumder, 2010, p. 2].

The military soon got the idea that a death ray would be handy in any battlefield and this lead to huge increase in research funding. Many of today's lasers were quickly developed during the 1960s hidden in research laboratories and military establishments [Steen and Mazumder, 2010, p. 3]. However something interesting happened in the beginning of the 1970s. Enthusiasts from the automotive industry realized the lasers potential for *material processing* [Ion, 2005, p. 1].

Actually, the first "demonstration" of laser material processing appeared in the 1964 film Goldfinger. The scene showed innovative thinking and industrialists quickly realized that ruby laser pulses might be suitable for drilling. Ruby laser welding were used in fabrication of the first Apollo lunar sample return containers in 1969. The first commercially available CNC soldering machine based on a 50 W CO₂ laser was produced in 1976 [Ion, 2005, p. 20].

In the early 1980s industrial CO₂ lasers with higher power and better reliability were built. Especially the development of the fibre-optic cable meant that higher powers could be transmitted without using complicated mirror systems. It also gave the possibility of mounting a laser on a robot [Ion, 2005, p. 25].

In the 1990s the CO₂ laser technology was still preferred for cutting and welding

¹Acronym for Light Amplification by Stimulated Emission of Radiation.

²Acronym for Microwave Amplification by Stimulated Emission of Radiation.

tasks. In 1997 a 4 kW CW Nd:YAG lasers was marketed, which provided direct competition with the leading CO₂ lasers. The automotive industry started replacing the CO₂ lasers with the Nd:YAG lasers for complex cutting and welding operations. At the end of the decade a 10 kW CW³ Nd:YAG⁴ unit was commercially available. Diode lasers were also investigated during the 1990s as a substitute for CO₂ and Nd:YAG. However there were problems with the thermal load and a lot of cooling were needed. In the late 1990s a 6 kW infra-red diode laser were commercially available [Jon, 2005, pp. 28-29].

Around the turn of the new millennium, compact and energy efficient diode-pumped solid-state lasers started to appear. This new type of lasers opened a whole new world of possibilities. They are small enough to mount directly onto robots and have better properties than lamp-pumped lasers all-round [Jon, 2005, pp. 31-32].

History have showed that the production industry have had a huge impact on the development of laser technology. Industries are good at realizing were there is a potential for developing and implementing new production technologies. This is also why it is very certain that new laser sources will continue to be developed and made more compact, efficient and cheaper. The ROBOCUT project can be considered an attempt in taking a large step forward in making processing more efficient, cheaper and more reliable. The project mainly aims at developing a laser cutting technology that will outperform the state-of-the-art laser cutting technologies. This thesis is a part of building the theoretical foundations for the ROBOCUT project.

1.1 The ROBOCUT Project

This section will present the basic idea behind the ROBOCUT project. This presentation will not go through the technical details as they will be mostly irrelevant for the remainder of the report. The main focus will be to explain the concept. For further technical information about the ROBOCUT project see [The Danish National Advanced Technology Foundation, 2010]. Furthermore the expected business potential of the project will be presented.

The vision of the ROBOCUT project is to develop a new laser cutting technology that will outperform state-of-the-art laser cutting. This will be made possible by effective Remote Laser Cutting (RLC), i.e. without assist gas. Because the process uses no assist gas, no mechanical device will have to follow the cutting point. The remote laser cutting technology relies on a multi-beam principle, see Figure 1.1(b). Traditional laser cutting is carried out with a single beam and assist gas.

³Continuous Wave

⁴Neodymium-Doped Yttrium Aluminium Garnet

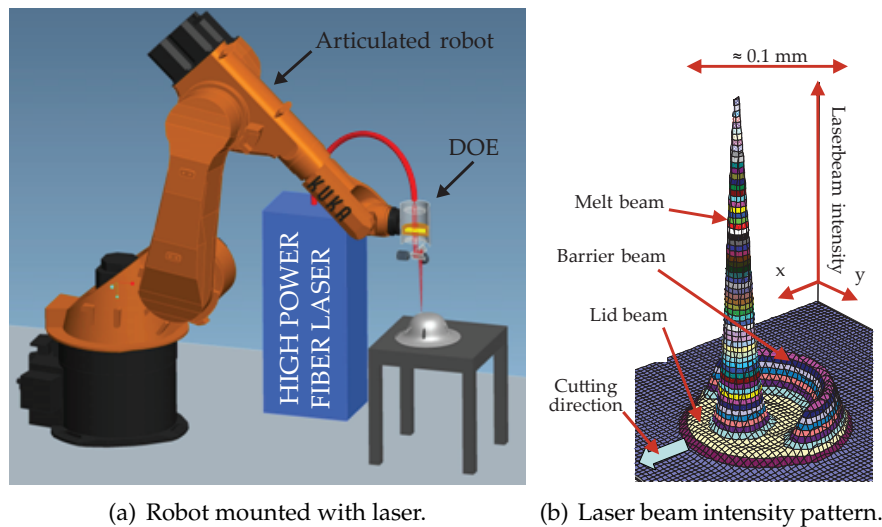


Figure 1.1: Illustration of ROBOCUT-concept [The Danish National Advanced Technology Foundation, 2010, p. 3, Figure 1].

Figure 1.1(b) shows the beam intensity pattern. The idea is to generate vapor pressure on the molten material at the most appropriate places in the kerf using a barrier beam. This will create an optimum melt flow away from the cut kerf. The simple approach of focussing a round beam to a small spot will not create an optimum melt flow without assist gas. When remote cutting with a simple round beam there will also be an uncontrolled melt ejection towards the laser beam.

The core of the ROBOCUT technology is therefore being able to construct a beam intensity pattern like the one shown in Figure 1.1(b). The beam originates from the fiber laser source and is passed through an optical cable to the scanner head consisting of an advanced optical system with a specially designed artificial hologram, also known as a *Diffractive Optical Element* (DOE), see Figure 1.2.

Besides the primary objective of developing a new RLC technology, a secondary goal of the ROBOCUT project is to study the use of the laser beam intensity pattern principle for improving the existing Remote Laser Welding (RLW).

The objective behind the ROBOCUT project is mainly to develop and integrate the ROBOCUT technology into the scanner head. The project also consists of other fundamental elements like establishing laboratory facilities, theoretical process simulation, simulation of flow patterns, optimizing laser beam pattern and so on. Currently the following projects are in progress:

- ◇ IPU and KUKA are developing the DOE. A PhD project is supporting this
- ◇ A PhD project is modeling the laser cutting process
- ◇ AAU is setting up laboratory facilities for testing

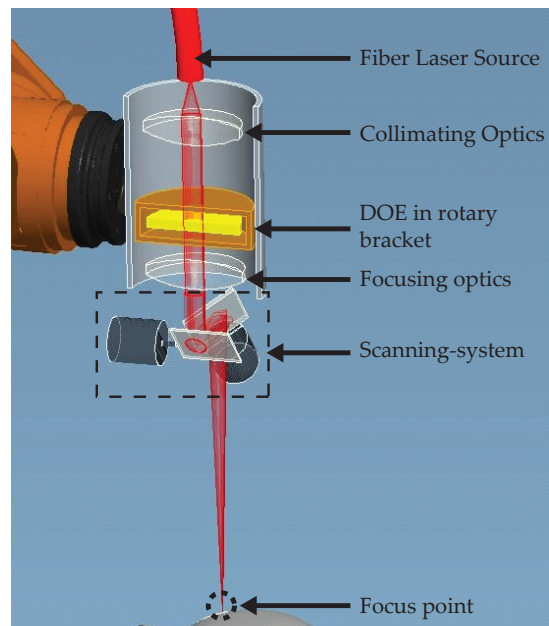


Figure 1.2: Close-up of the principle for the scanner head optics [Olsen, 2011, p. 1].

1.1.1 Business Potential

Laser cutting is a market that has been growing steadily for the last 30 years. The growth in the market has unfortunately been drastically reduced by the global crisis in 2008, as seen in Figure 1.3. The graph shows that the industrial laser market survived the dot-com recession of early 2000 almost without any downturn in revenues. The recession of 2008-2009 was however more global and devastating for the industrial laser market.

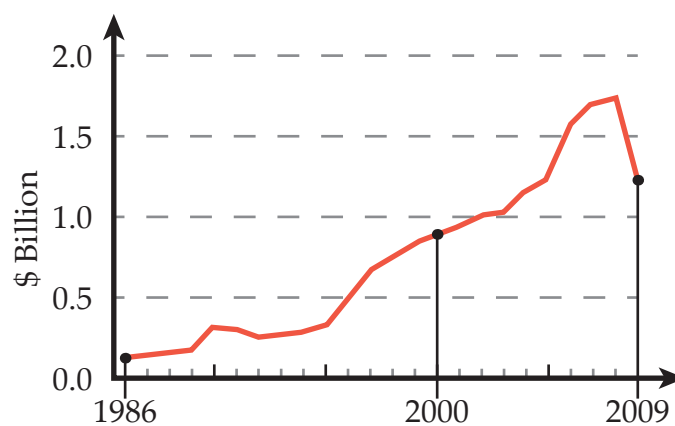


Figure 1.3: Industrial laser revenues - 1986-2009. [Belforte, 2010, p. 2, Figure 1].

Table 1.1 shows that no industrial laser category escaped the recession. Overall the revenues for all types of industrial lasers decreased 30% in 2008. This is a record for the industry [Belforte, 2010, p. 2]. However Table 1.1 also shows that the projected

revenue for 2010 shows an increase in the revenue. The CAGR⁵ for industrial laser revenues over the past 15 years is 12.8%. At this rate the revenue should return to the 2008 level by 2012.

Laser	2008	2009 EST	% Change	2010 Proj.	% Change
CO2	1091	669	-39	723	8
Solid State	394	340	-14	365	7
Fiber	213	169	-21	190	12
Other	60	53	-12	58	9
Total	1758	1231	-30	1336	9

Table 1.1: Global industrial laser revenues (\$ Million) [Belforte, 2010, p. 2, Table 1].

The growth in the industrial laser market is a consequence of the continuous improvement in productivity of the process. There is no indication that the development of laser processing is about to stall and laser cutting will take over more and more market shares from conventional sheet metal operations such as cutting, nibbling and punching. The ROBOCUT technology has the potential to outperform state-of-the-art laser cutting technology and thus totally dominate this market in the future. There is also a growing market for efficient 3-dimensional laser cutting in the car industry. On longer term, a market for high speed laser cutting in trimming of body components after press forming will also emerge [The Danish National Advanced Technology Foundation, 2010, pp. 4-5].

A thorough analysis of the potential revenue by the ROBOCUT technology has been made by the ROBOCUT stakeholders. Figure 1.4 shows the estimated revenue.

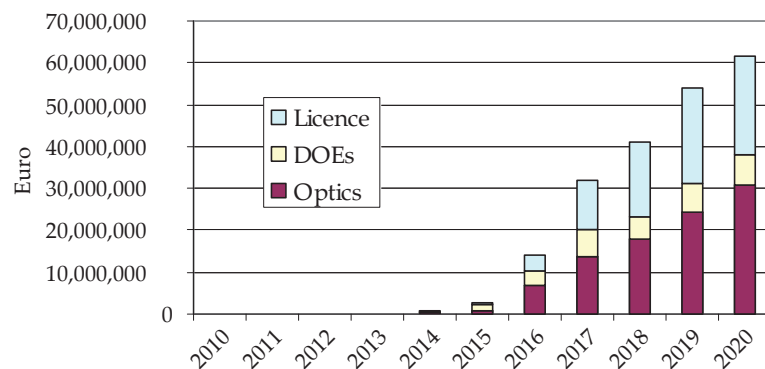


Figure 1.4: ROBOCUT annual revenue estimate [The Danish National Advanced Technology Foundation, 2010, p. 6, Figure 5].

As it can be seen, the revenue depends highly on the licence which means that obtaining a patent protection of the ROBOCUT-invention is crucial. It will also be important to ensure that competitors will respect the patent rights. However if

⁵Compound Annual Growth Rate

the ROBOCUT project succeeds in obtaining patent protection it has a great earning potential. The market has recovered from the crisis and will continue to grow for many years to come.

1.2 Scope of this Project

This thesis focuses on establishing some of the theoretical foundations of the ROBOCUT project. The thesis objectives are divided into two parts:

- ◇ Scheduling of the laser cutting process.
- ◇ Development of a ROBOCUT configurator.

The scope and content of each of these parts are presented below.

1.2.1 Part I: Scheduling of the Laser Cutting Process

One of the possible advantages of the new RLC technology is that it allows for programming any desired cutting operation without risking collisions between the laser optics, the workpiece or clamping devices. Because the detailed positioning and focusing of the laser beam is handled by the fast moving optical system in the scanner head, the RLC technology can also offer a significant decrease in cycle times when compared to conventional laser technology. At the same time the potential advantages of the technology is also its main challenge because of the added complexity in terms of planning and programming the process.

For this reason one of the main challenges of the ROBOCUT project is the effective transformation of cutting paths from CAD data into motions of the process equipment. Among others this entails a detailed path planning for the movement of the process equipment and generation of the necessary machine code. This is an area that have already been widely covered for conventional laser cutting, and as a result standard software is available for doing this. But because of the added complexity introduced by the RLC technology these are not directly transferable. For this reason it is necessary to carry out some research to identify some solutions for carrying out the path planning for RLC effectively.

An important part of path planning is to determine in which order the individual cuts should be carried out. This also known as the discipline of scheduling, which deals with the allocation of tasks to limited resources with the purpose of optimizing in terms of some objective(s). In this case the tasks consists of the cuts that needs to be carried out, while the resource is the RLC process and the objectives are related to the part and process requirements.

Part I of this thesis tries to identify some methods of scheduling the RLC process. This is carried out based on an example case at Ib Andresen Industri A/S. The reader is referred to Appendix A for a description of the company and their manufacturing processes.

1.2.2 Part II: Development of a ROBOCUT Configurator

The new RLC and RLW technology developed through the ROBOCUT project is widely applicable to many different production scenarios. This is also expressed through the very diverse application scenarios that the participating and supporting companies have in mind.

The fact that the RLC and RLW technology has such a wide application must be kept in mind during the development of the technology. That is, the technology must be developed to ensure that it is as generally applicable as possible, without compromising the cost and performance of the technology. This is a challenge, as the RLC and RLW technology is going to be comprised of a lot of subsystems working together to fulfill the overall expected functionality. At the same time, the needs and requirements for these subsystems will differ between different application scenarios. This raises the need for further examining the possible application scenarios with the objective of identifying the subsystems that possibly can make up the RLC and RLW process. Because the subsystems can be considered as separate modules, this also gives rise exploring the idea of developing a configurator for the RLC/RLW process. This is the subject of Part II.

Product configurators are used extensively for configuring products online. In the future we may instead imagine that configurators also are used for producers to buy production systems. Because the ROBOCUT technology has potential to be used in many production scenarios it may be a suited technology for such a production system configurator.

The objective is to develop a configurator that will clarify the capabilities of the technology in different production scenarios. The purpose of the configurator will be elaborated in Part II.

CHAPTER 2

Problem Formulation

This thesis is a part of the ROBOCUT project. The overall objective of this thesis is to help establish some of the theoretical foundations for the ROBOCUT project. Issues that needs research have been identified and this thesis will focus on some of those issues.

The project objectives can be divided into two elements with the following problems:

- ◇ Scheduling of the laser cutting process.
 - What methods are best for scheduling remote laser cutting and how can they be applied?
 - How can an interface be developed for defining a scheduling problem and visualizing scheduling results?
- ◇ Development of a ROBOCUT configurator.
 - What methodology can be used to prepare a production unit for configuration and how can it be applied to a remote laser cutting/welding production unit?
 - How can a configurator for a remote laser cutting/welding production unit be developed and deployed?

The structure of the thesis is divided into two parts. Part I focuses on the scheduling of the laser cutting process and Part II focuses on the development of the configurator.

PART
I
SCHEDULING SYSTEM

CHAPTER 3

Introduction

As previously mentioned, the vision of the ROBOCUT project is to develop a new Remote Laser Cutting (RLC) and Remote Laser Welding (RLW) technology, by utilizing a cutting head with focussing optics and scanning mirrors for rapid and precise positioning of a laser beam with a micro pattern. The technology has a great potential for increasing the flexibility of laser cutting while at the same time decreasing the cycle time, by offering faster laser beam positioning and cutting speeds than conventional laser cutting. To take full advantage of the technology however, it is important to ensure an effective planning and transformation of the cutting tasks into movements of the equipment.

Experience from robotic Remote Laser Welding (RLW) has shown that a lack of effective programming techniques of the equipment means that it is not profitable for lot sizes below 100.000 pieces per year [Hatwig et al., 2010, p. 327]. Thus, this is an area that still needs to be researched, to fully take advantage of both RLW and RLC technologies.

Before being able to program a laser cutting process it is necessary to plan and schedule the process. This is done by first identifying the cutting tasks (e.g. start point, end point, geometric data) along with the process parameters and constraints (e.g. cut speeds, laser power, max/min cut angle etc.). The identified cutting tasks then needs to be scheduled to obtain a feasible cut sequence, while satisfying the identified process parameters and constraints. The result from the planning and scheduling can then be used for programming the process equipment.

This part of the thesis will focuses on researching different techniques for the planning and scheduling of the cutting tasks for RLC. The research is conducted based on the industrial application of the RLC technology.

The following section presents a proposal for the composition of an integrated planning and scheduling system for the RLC process. This then leads to an introduction to scheduling before describing the definitions and notations used throughout the remainder of this part. Finally, the last Section presents some related work carried out in the field of RLC and RLW.

3.1 Integrated Planning and Scheduling

The planning and scheduling of the RLC process becomes a complex and daunting task for a human process planner even with just a few simple cutting shapes. For this reason a planning and scheduling system that can be fully integrated with a system for effectively programming the equipment is essential.

This section will present a proposal of a possible general system for programming the RLC process with integrated planning and scheduling. The proposed system is used as a means for identifying the inputs and outputs for the planning and scheduling. The system is shown using the Integrated Definition for Function Modeling (IDEF0) methodology [Federal Information Processing Standards, 1993].

The top level technical outside view of the RLC Program Generation System is illustrated by the A-0 diagram in Figure 3.1.

The system takes the geometric data of the part and the cutting tasks as an input in the form of a CAD data file. The RLC program generation system then transforms these data to some machine code for the equipment (e.g. cutting head, robot, etc.) and a visual illustration along with an estimated process time. To do this, the system utilizes a process planner (i.e. a human operator) along with some planning, scheduling and CAD/CAM software as the executing mechanisms. In order to ensure that the system outputs a feasible and valid solution for the RLC process, the system uses the process constraints as the controlling mechanism. A decomposition of the

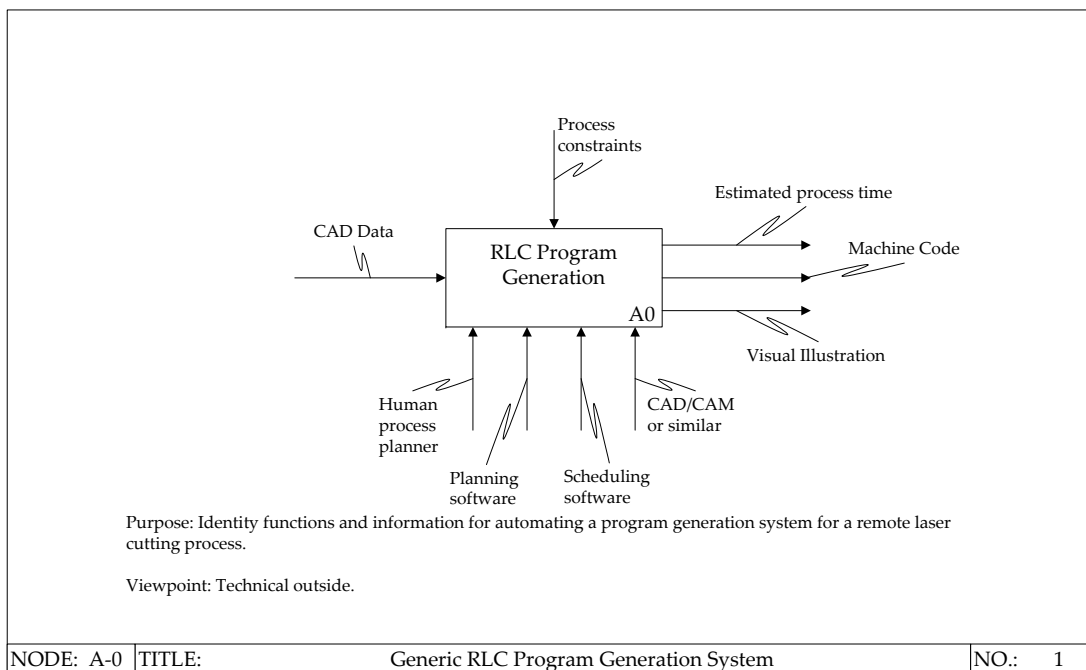


Figure 3.1: Top-level IDEF0 diagram of the scheduling system.

top-level technical outside by one level is shown in the A0 diagram in Figure 3.2. As it can be seen the system consists of four main functions.

In the first main function, *Process Planning*, the cutting process is planned by transforming the CAD data into separate cutting tasks and possible process parameters. The human process planner analyzes the part and the cut geometries based on experience and process constraints to determine if the cuts are possible to perform. Depending on the process setup and scenario, a software for aiding the process planning might be used, or in some cases perform the planning autonomously.

After the initial process planning, the second function, *Scheduling & Optimization*, the identified and sequenced cutting tasks are scheduled and optimized based on the process parameters and constraints. This function is carried out autonomously by a scheduling software. As an output, the function returns a scheduled task sequence and an estimated process time, the latter being a system output.

Based on the CAD-data, information about cutting tasks and the task sequence, the system is able to generate the appropriate programs for the process equipment in the third step, *Program Generation*. As the previous step, this step is also performed autonomously by a CAD/CAM¹ software or similar, with the process parameters and process constraints as the controlling mechanisms. The generated machine code is then returned as a system output, ready for transferring to the appropriate equipment.

In case simulation and/or visualization is desired for verification and communication purposes, this is handled by a fourth function, *Simulation & Visualization*. This function operates based on the same data as the *Program Generation* function, but also uses the generated machine code. As output the function returns some visual illustration of the results (e.g. graphs, animations, images). The software used for the simulation and/or illustration is also some CAD/CAM software, and in some cases the same software can be used for both program generation and simulation in which case these two functions essentially merges to one.

As it was mentioned earlier, this thesis does not focus on developing the entire RLC Program Generation System. Instead, the steps concerning the process planning and scheduling are mainly considered. The last step concerning the simulation and visualization of the results will also receive some attention, but only in terms of illustrating and verifying the obtained scheduling results.

¹CAM: Computer Aided Manufacturing

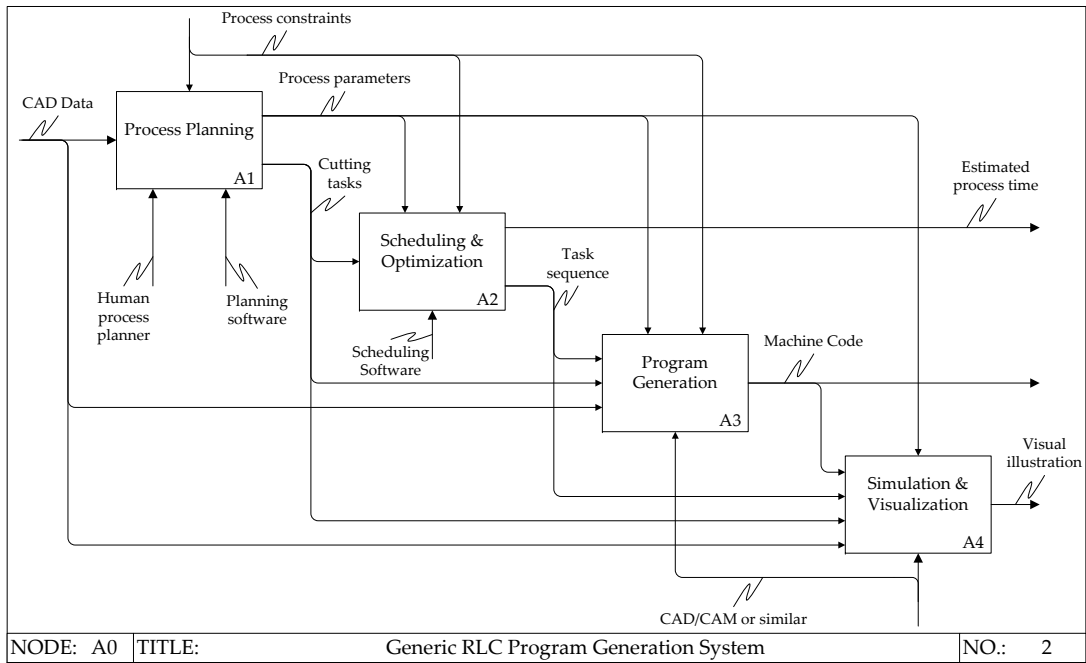


Figure 3.2: IDEF0 A0 diagram showing an overview of the complete scheduling system.

3.2 Introduction to Scheduling

The discipline of scheduling deals with the allocation of limited resources to the processing of tasks with the purpose of optimizing in terms of one or several objectives. Examples of objectives are to minimize the cycle time, lateness or idle-time [Pinedo and Chao, 1999, p. 2].

The discipline of scheduling is used in a wide range of manufacturing and service industries. The main purpose of scheduling is to minimize the production time and costs. This is done by calculating when to make what, with which staff and on what equipment. In other words scheduling aims to develop the best schedule while satisfying certain timing and sequence conditions.

This is a very general definition of scheduling and it should be noted that scheduling is applicable to a wide range of problems. Some common scheduling applications are [Pinedo and Chao, 1999, pp. 2-5]:

An auto-mobile assembly line. The objective is to maximize throughput by sequencing the cars in a way that balances the workload at each station over time.

A production plan for a paper mill. The objective is to maximize production while minimizing inventory costs.

A reservation system. The objective is to maximize the number of days the cars are rented out.

Scheduling nurses in a hospital. The objective is to develop shifts assignments that meet all daily requirements and satisfy the constraints at a minimal cost.

The examples underline how used and important scheduling is. Scheduling will often have a noticeable impact on system performance. Even though the examples give the impression that scheduling are used for developing production plans or developing shifts assignments it can also be used for process optimization. This part of the thesis deals with such a process optimization - scheduling of the laser cutting process.

Scheduling of the laser cutting process will aim at sequencing and timing individual parts of a cutting job. The problem is to develop an algorithm or heuristic that will effectively schedule the individual tasks while ensuring that no constraints are violated. This problem is complicated by the new process capabilities in remote laser cutting.

Scheduling can be difficult to perform and implement. The technical difficulties resembles those from combinatorial optimization. The problems are generally related to the modelling of the real-world scheduling problems and the retrieval and management of information. Even though it is difficult to overcome these problems it is often worth the effort in terms of increased efficiency.

3.3 Scheduling Definitions and Notations

Typically planning and scheduling operates on the two levels: *Production planning and master scheduling* and *shop order planning and scheduling*. These operates with medium- to long-term planning for an entire organization and short-term planning and scheduling of the production line, respectively [Kolakowska, 2010, p. 8]. In the latter case, the theories of scheduling are applied for the allocation of limited resources to the processing of *jobs*, where a job is defined as a single operation or a set of operations [Pinedo and Chao, 1999, p. 12]. Typical examples of this may be allocation of packages to delivery vehicles at a shipping company, allocation of rooms for lectures at a university, or jobs to machines in a workshop [Pinedo and Chao, 1999, p. 2].

As mentioned previously this thesis deals with process planning and scheduling, which is a lower level than the above mentioned examples. In turn, this means that the scheduling problem concerns the allocation of *tasks* (set of operations) of a single job on a particular machine (i.e. the RLC head). Based on this, the following definitions and notations will be used throughout the remainder of this part:

A cutting job, J , is defined as a single part containing a set of cuts.

A cutting task, j , is defined as a geometry with connected cutting paths. This means that the cutting of a hole, a single line (i.e. no connected cut paths), and a series of connected paths are defined as individual cutting tasks. The reason for this is due to the difference in cutting angle (as described in section B.1) when cutting while the cutting head and/or part moves relative to each other.

Processing time, P_j , is defined as the time that cutting task j takes to cut.

Release date, RD_j , is the time at which cutting task j is released for cutting. That is, the earliest time at which the cutting task can be commenced.

Due date, DD_j , is the time at which cutting task j needs to be finished.

Completion time, C_j , is the time at which task j has been completed. Identifying the maximum completion time $C_{\max} = \max(C_1, \dots, C_j)$ will yield the time at which all the cutting tasks have been completed. Put in other words, C_{\max} is the time of completion of the last cutting task, and is therefore also known as the *makespan* [Pinedo, 2009, p. 28].

3.4 Related Work

Even though the technological idea behind the RLC and RLW technology developed through the ROBOCUT project is new, the idea of RLC and RLW is not, as seen in Tahmouch et al. [1997] and Antonova et al. [2000]. In addition remote laser welding systems and scanner heads are already available, like the KS Roboscan remote laser welding scanner head available by KUKA [Kuka Systems GmbH, 2011]. For these reasons some work has already been carried out in relation to the scheduling of RLW and RLC, although it is has only been possible to find a limited amount, some of which will be presented here.

One of the main challenges with the application of the RLC and RLW processed is the lack of sufficiently effective programming methods. The work done in this area has primarily focused on the programming of robot based RLW and RLC. To ensure a cost-effective programming of these systems it should be carried out as autonomously as possible, however this has proved difficult because of the kinematic redundancy that is introduced by the scanner head. To overcome this problem, Stemmann and Zunke [2006] successfully studied the use of the Generalized Traveling Salesman Problem (GTSP) (introduced by Srivastava et al. [1969]). In short this approach seeks to subdivide the welding tasks into smaller clusters (or working areas), and then find the shortest possible route through them. Each of these clusters is composed of a set of points, and exactly one has to be visited.

A similar approach is used by Reinhart et al. [2008], where the problem is also

divided into subtasks first and then afterwards the optimal path between the subtasks is found using the conventional Traveling Salesman Problem (TSP) (described later in Section 6.2). In this paper they also propose the use of augmented reality as a tool for visually showing the resulting paths onto the part itself by projection. Later on this approach has been extended by Hatwig et al. [2010] to form a complete concept for automated path planning of robotic RLW and RLC.

Instead of focussing on the task and path planning with the objective of obtaining the shortest possible path, Zaeh et al. [2010] describes another possible objective. This objective concerns the minimization of thermal distortions of the part that is cut or welded, by changing the sequence of the task in terms of their thermal contribution. This is possible within the working area of the scanner head without a significant reduction of the cycle time, because of the fast positioning and focussing of the laser beam by the scanner head. A test implementation of this approach showed a significant decrease in part distortions, and thus marks another advantage facilitated by the RLW and RLC technology, that is made possible through careful path planning and scheduling.

CHAPTER 4

The Scheduling Problem

The RLC technology is applicable to a number of different production scenarios. Depending on the scenario, the process requirements and constraints changes accordingly. In effect, the planning and scheduling problem also changes according to the scenario. As a result, it is not possible to develop a generic planning and scheduling of the RLC process. For this reason the planning and scheduling of the RLC process will be conducted based on a specific scenario: RLC in the roll forming lines at Ib Andresen Industri A/S (IAI), which is described in Appendix A.1. The knowledge and experience obtained through this scenario can then be used for other possible usage scenarios.

The following will provide an introduction to the general scheduling problem of the RLC process, by first presenting the possible usage scenarios along with some scheduling related considerations, followed by a description of the process constraints. Section 4.2 presents the scenario that is considered for scheduling in this thesis. After this the scheduling problem is mathematically described along with a summary of assumptions. Section 3.4 comments on related work in this area.

4.1 Usage Scenarios

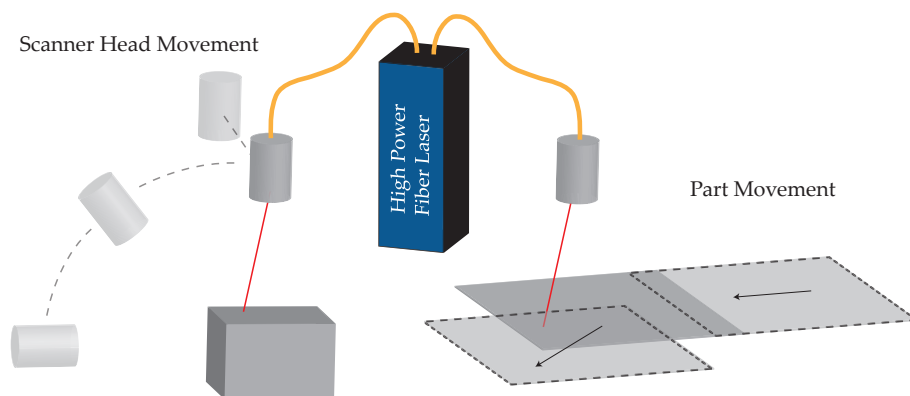


Figure 4.1: Illustration of the possible usage scenarios. It is possible to move the scanner head relative to the part, move the part relative to the scanner head or move both. Note that a laser will only feed a single scanner head - two are shown here for illustration purposes.

As it was described in Section 1.1, the RLC process will utilize a scanner head with focussing optics and scanning mirrors for precise positioning of the laser beam onto the parts to cut. As it is also illustrated by Figure 4.1, this makes the scanner head capable of cutting shapes from a static position. At the same time, the scanner head offers increased flexibility for cutting operations while moving. In addition, the parts to be cut may also be both static and moving relative to the scanner head. Hence the possible usage scenarios can be divided into the main scenarios:

Static Scanner Head, *where the head is mounted in a static position above the part to cut.*

Static parts: This is the scenario all the holes to cut can be contained within the possible cutting area of the scanner head. This could be the case when cutting smaller parts (like the workpiece shown in Figure 1.1), where an ordinary die press could prove infeasible. In terms of scheduling, this is the simplest case, as the scheduling problem basically reduces to finding the cutting sequence with the shortest possible processing time.

Moving parts: If the holes cannot be confined within the possible cutting area, if the parts need to be cut while moving continuously or for other process requirement reasons (e.g. cutting angles, part interference, etc.) it might be necessary to move the part relatively to the scanner head (the right part of Figure 4.1). The cutting of holes in the metal strip before roll forming line is an example of this. This is also the example considered in this thesis. In these cases the scheduling problem could include objectives like minimizing the process time and part movement, while satisfying part requirement constraints (see section B.1). In the case of a continuously moving part, the holes also needs to be cut within a certain time frame. Also, the possible move speed and acceleration of the part needs to be taken into account.

Moving Scanner Head, *where the head is mounted on a robotic manipulator or similar for moving the scanner head relative to the part.*

Static parts: In many cases it might be easier to move the scanner head relative to the part, instead of moving the part. For instance when laser cutting car parts, or when laser cutting in large metal sheets (like the existing CNC laser cutting process). The scheduling of these cases is almost identical to that of a static head and moving part, only in this case the speed and accelerations that needs to be considered are those of the scanner head.

Moving parts: If the parts to laser cut are so large that a robot is not able to reach all of the cutting areas, the part might need to move as well. This could be the case when laser cutting car bodies as they move down the production line. In

some cases the part might have to move continuously, even though it is possible to reach all cutting areas. This is the most complex scheduling problem, the scheduler needs to take the movement, velocities, accelerations, and possible interferences of both the part and the scanner head into account, while satisfying the part requirements constraints as well as objectives like minimizing the process time and/or total movement, etc.

4.2 Scheduling the RLC Process for Roll Forming

Because the RLC technology is applicable to different scenarios, the process planning and scheduling will be conducted based on the industrial application of the RLC technology in the roll forming lines at IAI. A description of roll forming and the potential of using RLC in the roll forming process is given in Appendix A.

The scenario considered is based on a typical part produced by the roll forming process (Figure A.5). Instead of pre-punching the holes, this scenario assumes that the holes will be cut using the RLC process instead. Thus, the usage scenario considered is the *Static scanner head - Moving parts* (as described in section 4.1). The resulting cutting process is illustrated by Figure 4.2.

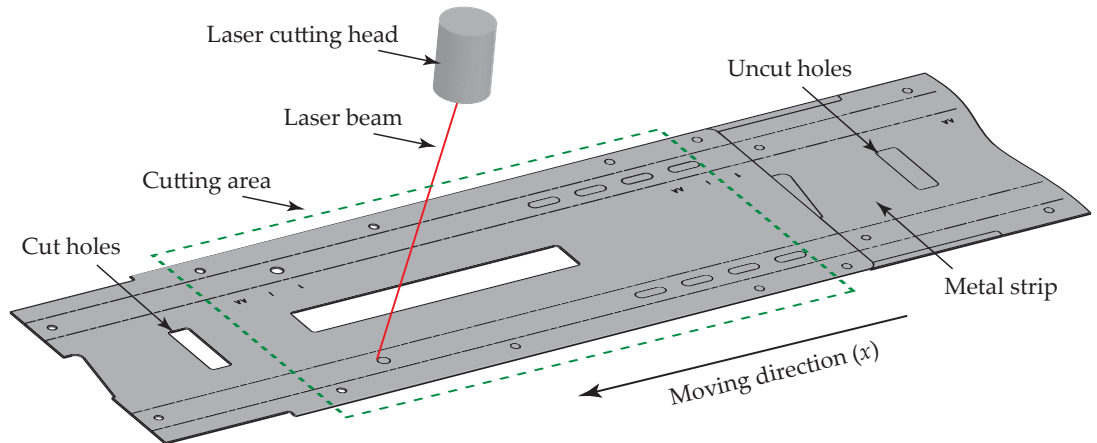


Figure 4.2: The RLC process with static mounted scanner head for cutting holes in a metal strip prior to roll forming.

The scanner head is placed in a fixed position above the moving metal strip. The metal strip passes underneath the scanner head at a constant velocity, while the holes are cut on the fly by the laser beam. *The planning and scheduling problem thus consists of transforming the holes into cutting tasks and a feasible solution that can be used for programming the scanner head.* IAI has expressed that they would want to know at what maximum velocity the part may move in order for it to be possible to laser cut on the fly.

Since the case considered deals with the pre-cutting of holes ahead of roll forming, the first step towards planning and scheduling is to ensure that cutting the holes will not cause any problems (described in Section A.2), during the roll forming process.

As the laser beam is only able to (or restricted to) a certain cutting area, the individual holes needs to be cut within a certain *time window*, that is the time from which the hole enters the cutting area to the time the hole exits the cutting area. In addition the process time should also be minimized, since a faster processing time for cutting holes in turn increases the possible conveyor speed of the metal strip, and thus the total throughput of the roll forming mill. The two time constraints depend on both the *conveyor speed* of the metal strip, the *cutting speed* and the *move speed* of the laser beam by the scanning mirrors in the scanner head.

While satisfying the time constraints described above, the scheduling might also need to take the cutting angles into account. In case a part has requirements for a maximum and/or minimum cutting angle this constraint needs to be applied. When cutting in thick sheet metal strips, it might be necessary to apply a constraint concerning the maximum cutting angle as well. For a detailed description of these and other possible constraints, the reader is referred to Appendix B.1.

4.3 Mathematical Description of the Problem

This section focuses on describing the problem of optimizing the time it takes to laser cut a part. The actual cutting time cannot be optimized but the time the beam needs to move from one point to another while not cutting can be minimized.

In order for it to be possible to describe this optimization problem mathematically some assumptions are made. First of all focus is on laser cutting in two dimensions. This is equivalent to laser cutting in sheet metal.

A cutting job is divided into a series of tasks. A task is defined to be a line segment or closed shape. Figure 4.3 shows an example of a cutting job where the grey dashed lines is indicating what needs to be cut. In the Figure the star is a task, the ellipsis is a task and so on. The following assumption is now made: When the laser starts cutting a task it will finish cutting this task before moving on to another task.

Each task has a predefined point where the cutting starts (and ends if it is a closed geometry). At the moment the location of this point is not commented. An example of the current optimization problem can be illustrated using Figure 4.3 where the arrows illustrates how the laser moves from one task to another while not cutting.

The start point for each task is predefined and thus the combinatorial problem is reduced to finding the shortest route between point one, two, three and four for the

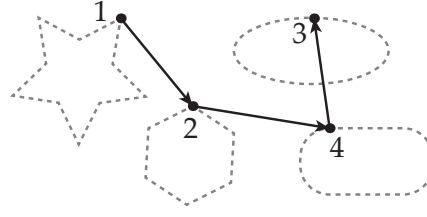


Figure 4.3: Illustration of random laser cutting job.

example job in Figure 4.3. In a more mathematical sense the optimum path is given by ordering the points p_1, \dots, p_n such that $\sum_{i=1}^{n-1} d(p_i, p_{i+1})$ is minimum, where d is the distance between p_i and p_{i+1} . For two points $p = (x, y)$ and $p' = (x', y')$ in the plane the distance between them is of course $d(p, p') := \sqrt{(x-x')^2 + (y-y')^2}$. An order of the points can be represented by a permutation, i.e. a bijection $\pi : \{1, \dots, n\} \rightarrow \{1, \dots, n\}$.

Which permutation is best of course depends on the placement of points and other constraints. One instance of the problem can however be considered a list of points in the plane, i.e. the coordinates of points. Thus the problem can be stated mathematically as follows (inspired by [Korte and Vygen, 2002, p. 1]) :

Laser Cutting Problem

Instance: A set of points $p_1, \dots, p_n \in \mathbb{R}^2$.

Task: Find a permutation $\pi : \{1, \dots, n\} \rightarrow \{1, \dots, n\}$ such that $\sum_{i=1}^{n-1} d(p_{\pi(i)}, p_{\pi(i+1)})$ is minimum.

The problem is now formulated mathematically and the task is to find a permutation such that the distance travelled is minimized. By making this mathematical description it is realized, that in order to minimize the total process *time* for the laser cutter, the *distance* travelled must be minimized. Remember that this statement is valid under the assumptions made in the beginning of this section. If we were considering cutting in three dimensions minimizing the distance travelled may not minimize the process time.

4.4 Assumptions and Approach

A number of assumptions has been made in order to state the laser cutting problem mathematically. This section will summarize these assumptions.

The assumption are listed in a bulleted list. Each bullet point identifies an important assumption. The sub bullet points identifies the assumptions on which the important assumptions depend.

- ◇ A cutting task can be represented with a process time and a point.
 - Once a cutting task is started it is finished before moving on to another task.
 - All tasks are closed geometries are therefore the task starting point is also the task end point.
- ◇ Minimizing the distance travelled is the same as minimizing the process time.
 - The cutting speed and movement speed is constant. Acceleration is not considered.
 - Cutting is done only in two dimensions
- ◇ The part moves it moves at a constant speed.

4.4.1 Defining the Cutting Tasks

The start points for tasks are, for the sake of implementation simplicity, always placed in the direction of the part movement. This is illustrated by Figure 4.4 where an example of a scheduled path is shown on the example part. Figure 4.4 also shows that tasks are numbered in the opposite direction of the part movement direction.

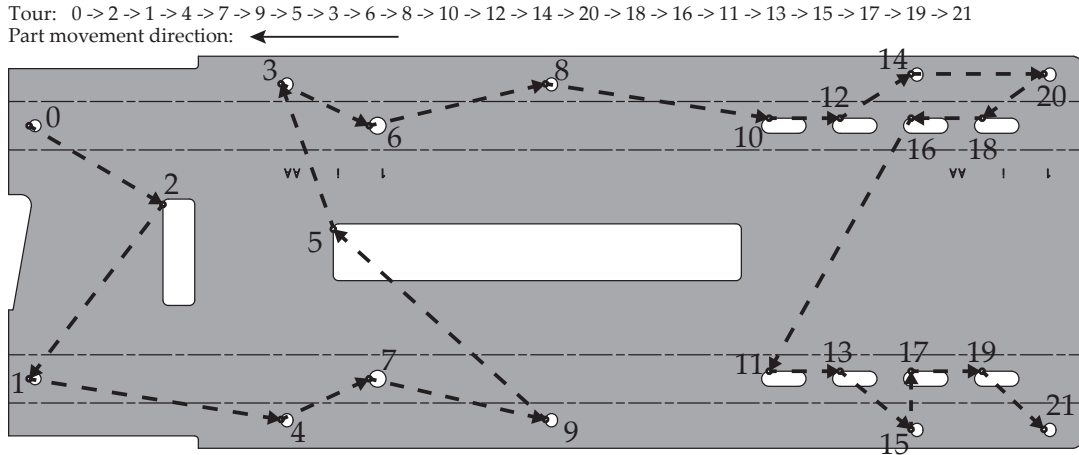


Figure 4.4: Illustration of scheduled path.

Each of these points are associated with their respective process times, defined by:

$$P_j = \frac{Cl_j}{v_{cut}} \quad (4.1)$$

Where Cl_j is the length of the cut path of task j and v_{cut} is the cutting speed.

4.4.2 Due Date and Release Date

As previously mentioned, each of the cutting tasks needs to be executed within a certain time window. In addition it is only possible to cut a task when it is fully inside the cutting area (see Figure 4.2). To take this into account each cutting task is associated with a due date and a release date. Figure 4.5 shows how they are calculated.

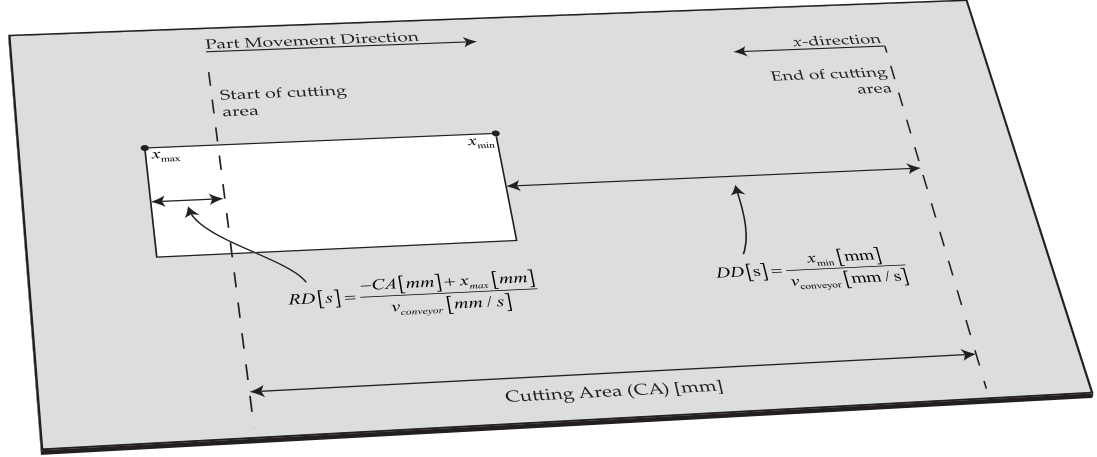


Figure 4.5: Illustration of cutting area along with Due Date and Release Date.

The due date, DD , and release date, RD , are calculated based on the placement of the tasks at the time $t = 0$. Because of the assumption of a constant conveyor speed it is only necessary to calculate the due date and release date once. If the conveyor speed was varying the due and release would need to be calculated at other time instances as well.

In Figure 4.5 the term x_{min} , used to calculate the due date, is equal to the x -value of the start point of the hole (task), while the term x_{max} , used to calculate the release date, denotes the x -value of the opposite end of the hole (task). To better understand the calculation, consider the following example. A cutting task located at $x_{min} = 500\text{mm}$ with the hole ending at $x_{max} = 700\text{mm}$, a cutting area of $CA = 600\text{mm}$ and a conveyor speed of $v_{conveyor} = 50\text{mm/s}$ yields the following results:

$$DD[s] = \frac{500\text{mm}}{50\text{mm/s}} = 10\text{s} \quad (4.2)$$

$$RD[s] = \frac{-600\text{mm} + 700\text{mm}}{50\text{mm/s}} = 2\text{s} \quad (4.3)$$

The result of Equation 4.2 and 4.3 should be interpreted this way: From the time the job is started ($t = 0$), the task must be finished after ten seconds and the cutting cannot start until after two seconds has passed. In the case that $x_{max} < CA$ the release date yields a negative result. This simply indicates that the task is inside the cutting area at $t = 0$.

CHAPTER 5

Dispatching Rule Scheduler

In some cases, a scheduling problem may be easily and efficiently solved using existing algorithms [Pinedo and Chao, 1999, p. 28]. When this is the case, the problem can be solved in *polynomial time*, meaning that even large instances of the problem can be solved in a reasonable amount of computation time. Of course this all depends on the particular objectives and constraints of the scheduling problem.

Unfortunately, this is rarely the case, as optimal solutions often cannot be found within polynomial time. In these cases the optimal solution may be found only by a complete enumeration of all the possible solutions of the problem. These scheduling problems, referred to as *NP-Hard* problems, requires a large amount of computer time that increases exponentially with the number of tasks that is to be scheduled [Pinedo and Chao, 1999, p. 28].

For small instances of a scheduling problem this may not impose a problem. However for large instances the computation time involved becomes so significant that it may not be available. To overcome this problem, some general purpose methods are used instead to find a feasible solution that is acceptable, but not necessarily ensures optimality. In return, these methods requires a significantly smaller amount of computation time, thus making them applicable to much larger problem instances.

In this chapter, some of the general purpose methods will be applied to the RLC scheduling problem. This is done with the objective of studying the applicability and usability of the schedules obtained by the faster general purpose methods. In the following section, some *basic dispatching rules* are explained and applied to the RLC scheduling problem. Section 5.2 then combines some of these into *composite dispatching rules* which is then applied to the RLC scheduling problem. Finally, the results obtained by dispatching rules are summarized and discussed in Section 5.3.

The MATLAB script for the implementations in this chapter can be found in the folder *Part I/Basic Dispatching Rules (MatLab)* on the enclosed CD.

5.1 Basic Dispatching Rules

A basic dispatching rule is a rule that is used to select the next job from a set of jobs waiting for processing on a machine. This is done by prioritizing each of the jobs according to some defined attributes. These may be attributes of the particular job, machine, or time depending on the objective(s) of the rule [Pinedo and Chao, 1999, p. 29]. In the RLC scheduling problem, the dispatching rules are used to prioritize and select the individual cutting tasks of a part for remote laser cutting. For more information about the rules, the reader is referred to [Pinedo and Chao, 1999, pp. 30-31]

In what follows a three basic dispatching rules will be applied to the RLC problem that was introduced in Section 4.2. To test the results obtained through the dispatching rules, they are tested using the parameters shown in Table 5.1 for the RLC process.

Remote laser cutting	
Cutting speed	30 m/min = 500 mm/s
Movement speed	470 m/min = 7833 mm/s

Table 5.1: Process data used for testing the dispatching rules.

Notice that the conveyor speed is not listed in Table 5.1 since this parameter is actually a variable. While the possible cutting and movement speeds depend on the RLC process and scanner head capabilities respectively, the possible conveyor speed will depend on the scheduled sequence. Thus, an additional objective of scheduling with dispatching rules, is to find the maximum possible conveyor speed.

First the *Earliest Due Date* (EDD) and *Minimum Slack* (MS) rule that has the maximum lateness as an objective are applied. Next, the SST rule will be applied, as setup time between cutting tasks (i.e. positioning of the mirrors) depends on the order at which the cutting tasks are scheduled.

5.1.1 Earliest Due Date First

As the name implies, this dispatching rule simply chooses the task in the queue that has Earliest Due Date, with the objective of minimizing the maximum lateness of the tasks, which is defined by [Pinedo, 2009, p. 29]:

$$L_{\max} = \max(L_1, \dots, L_j) \quad (5.1)$$

where L_j is the lateness of task j , given by:

$$L_j = C_j - DD_j \quad (5.2)$$

The rule is applied to the RLC scheduling problem by calculating the due dates for the individual cutting tasks (i.e. holes) of the part based on their position and the travel direction and speed of the metal strip. The resulting cut path from the EDD rule is seen in Figure 5.1.

Travel distance: 3242 mm

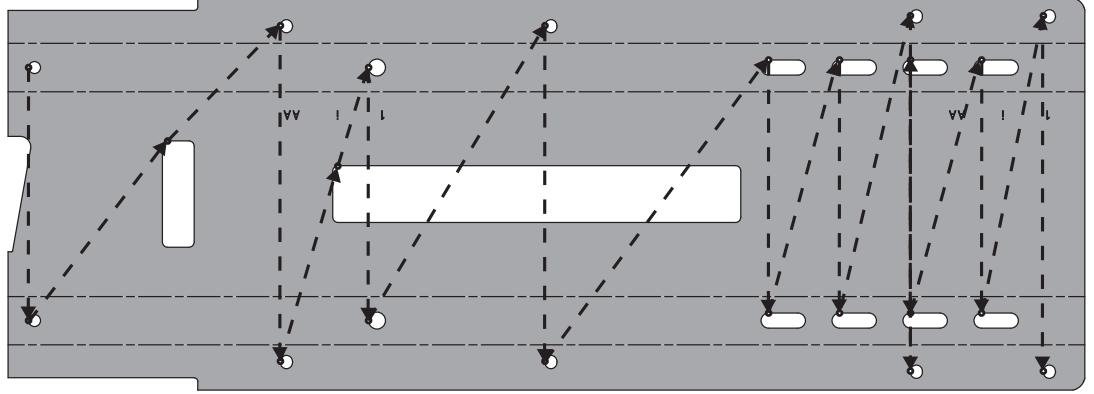


Figure 5.1: Resulting cut path for EDD rule with a conveyor speed of 180 mm/s.

In this case, the rule has been applied a conveyor speed of 180 mm/s. Changing the conveyor speed however will not affect the result as this will simply scale all due dates by the same factor.

As it can be observed, the EDD rule returns a schedule that forces the laser to travel back and forth between the holes on each side. Even though this is a valid schedule, it will most likely not yield the shortest cycle time. Based on the scheduled path, the total cycle-time is 3278 ms, which is equivalent to a maximum possible conveyor speed of around 185 mm/s.

5.1.2 Minimum Slack First

The *Minimum Slack* (MS) rule also has the objective of minimizing the lateness of the tasks (Equation 5.1, like the EDD rule. However, while the EDD rule only considers the due date of the tasks, the MS rule also takes the processing time into account by prioritizing the tasks according to the *slack* of each task, i.e. time until the task needs to be started in order to finish on time. Thus, the MS uses the following expression

[Pinedo and Chao, 1999, p. 30]:

$$S_j = \max(DD_j - P_j - t, 0) \quad (5.3)$$

Where t is the current time and S_j is the slack, DD_j is the due date and P_j is the processing time of task j . Equation 5.3 is the conventional version of the MS rule. When a task needs to be started at exactly time t or it is late, the MS rule will return its minimum value of zero. This can actually impose a problem if two (or more) tasks are late, in which case they both will have zero slack. In this situation the order of the tasks will have to be chosen either according to some other criteria or randomly, even though one of the tasks might be more late than the other. For this reason, the MS rule is modified to allow for a negative slack.

$$S_j = DD_j - P_j - t \quad (5.4)$$

This simple modification means that tasks are only chosen randomly when they have an equal slack (or negative slack), hence giving priority to the jobs that are the most late.

The best obtainable path (i.e. the fastest possible), by applying the modified MS rule to the scheduling problem, is shown in Figure 5.2.

Travel distance: 3405 mm

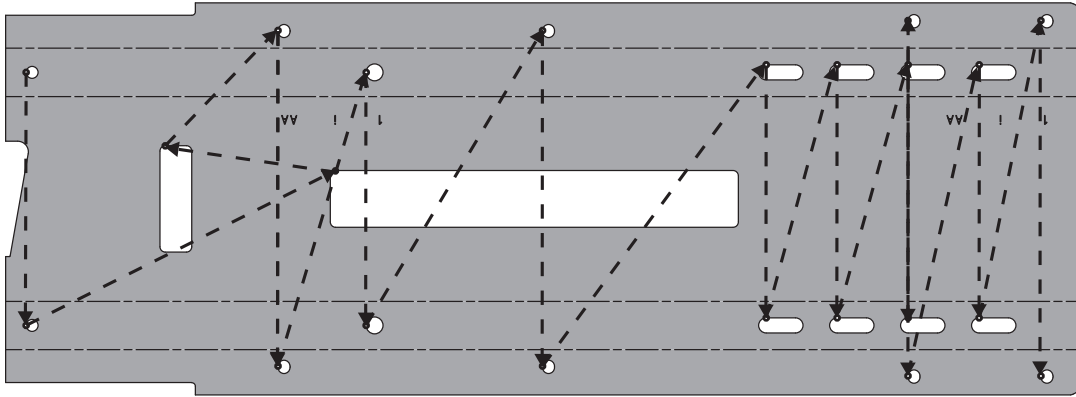


Figure 5.2: Resulting cut path for MS rule with a conveyor speed of 172 mm/s.

Comparing the path obtained using the MS rule, with that of the EDD rule in Figure 5.1 it is easily observed that only the first part of the paths differ from each other. The reason is that the cutting tasks with the largest processing time are among the first tasks, while the remaining tasks are similar in processing times. This also demonstrates the characteristic of the MS rule, that when tasks have equal due dates and processing times it basically reduces to the EDD rule.

The cycle time for the scheduled path shown in Figure 5.2 is 3300ms, equivalent of

a maximum conveyor speed of 184 mm/s which is about the same as the EDD rule. However, in practice the conveyor speed will be limited to 172 mm/s when applying the MS rule. The reason for this is that the task pertaining to the large center hole is scheduled as the first task when the conveyor speed is above 172 mm/s. Because of its long processing time it will cause the two tasks concerning the two small holes on the left to be delayed and unable to finish in time.

Contrary to the EDD rule, the path obtained using the MS rule will change as the cutting speed and conveyor speed is changed. By considering equation 5.4 it can be deducted that as the cutting speed is increased (decreasing the processing times) or the conveyor speed is lowered (extending the due dates) the MS rule approaches the EDD rule. This is considered to be an advantage, as this rule adds a dynamic behavior to the scheduling problem, thus changing the schedule according to the tightness of the due dates. More about this follows in section 5.2.1.

5.1.3 Shortest Setup Time First

Each time a specific cutting task has finished, the laser beam needs to be moved (repositioned) some distance to the next cutting task. Obviously this operation takes some time, and as no cutting is carried out during this time, it can be considered as a setup time. Furthermore, since the tasks are not equally spaced the total setup time will depend on the sequence of the tasks. This is also known as *sequence-dependent setup times* [Pinedo and Chao, 1999, p. 34].

Since the objective of the scheduling problem, described in Section 4.3, is to find a valid solution with the minimum total distance traveled, the *Shortest Setup Time* (SST) rule is applied next to take the sequence-dependent setups into account. As the name implies this rule selects the next task based on which has the minimum setup time, with the objective of minimizing the makespan, C_{\max} (see Section 3.3). Because the setups are sequence dependent, it is necessary to calculate a setup matrix, like the example shown in Table 5.2.

		To task:				
		1	2	3	...	n
From task:	0	2	4	1	...	s_{0n}
	1	-	5	2	...	s_{1n}
	2	5	-	2	...	s_{2n}
	\vdots	\vdots	\vdots	\vdots	\ddots	
	n	s_{n1}	s_{n2}	s_{n3}		s_{nn}

Table 5.2: Sequence-dependent setup matrix.

The sequence-dependent setup matrix shows the setup time, s_{lj} necessary when going from the previously executed task, denoted by l , to the next task, denoted by j .

moving- and conveyor speeds. Even though both of these rules provided some reasonable and valid solutions they produced a path that was not smooth and seemed to have a high cost (total travel length).

By taking the sequence-dependent setups into account, the SST rule provided a means of obtaining a path for the makespan objective. This yielded the fastest solution path in terms of the cycle time, while also providing a smoother and lower cost path than the EDD and MS rules. The tradeoff for this success however was problems with finishing all of the tasks before their designated due dates.

In summary the paths obtained by focussing on the two separate objectives each offer some advantages and disadvantages. In an effort to obtain even better results, the next section will study some methods for combining the basic dispatching rules into some dispatching rules with multiple objectives.

5.2 Composite Dispatching Rules

In many cases, including the current scheduling problem, there are more than one objective for the scheduling problem. In such cases the basic dispatching rules, which are only able to satisfy a single objective at a time, are not useful. Instead, some more elaborate rules which are able to take several parameters and objectives into account in order to find a reasonable schedule are needed.

One way of obtaining these more complex rules is to combine the basic dispatching rules to form what is known as *composite dispatching rules*. In a composite dispatching rule each of the basic rules that it is composed of is given its own scaling parameter that determines its share of contribution to the total dispatching rule expression [Pinedo, 2009, p. 445]. The exact composition and structure of the composite dispatching rules depend on the nature and type of scheduling problem, and this is also an area which has been researched [Pinedo and Chao, 1999, p. 32].

The following section will study and apply one such composite dispatching rule to the scheduling problem in an attempt to obtain a better solution than possible with basic dispatching rules. Note that the composite dispatching rules will also be applied using the same process parameters from Table 5.1, as well as the objective of finding the fastest possible conveyor speed.

5.2.1 Apparent Tardiness Cost with Sequence Dependent Setups

In this case the composite dispatching rule *Apparent Tardiness Cost with Sequence Dependent Setups* (ATCS) is studied further. This rule is used, as this offers a combination between the MS and SST rules that both showed some good results and

characteristics in terms of their respective objective functions. In addition, the ATCS rule also includes the *Weighted Shortest Processing Time* (WSPT) rule that provides the ability to prioritize the individual tasks in terms of their processing times by giving them weights [Pinedo and Chao, 1999, p. 30].

The ATCS rule uses the following expression to calculate a priority index for each of the remaining tasks [Lee et al., 1997, p. 46]:

$$I_j(t, l) = \underbrace{\frac{w_j}{P_j}}_{\text{WSPT}} \exp \left(\underbrace{-\frac{\max(DD_j - P_j - t, 0)}{K_1 \bar{P}}}_{\text{MS}} \right) \exp \left(\underbrace{-\frac{s_{jl}}{K_2 \bar{s}}}_{\text{SST}} \right) \quad (5.5)$$

Where j is the task index, l is the index of the task just executed, w_j is the task weight, K_1 and K_2 are scaling or look-ahead parameters, s_{lj} is the sequence-dependent setup time (from Section 5.1.3), \bar{P} is the average processing time and \bar{s} is the average setup time.

As it can be seen, Equation 5.5 is composed of the WSPT, MS and SST rules. The WSPT term favors the tasks with a high weight and low processing time. In this case however all of the tasks will receive an equal weight of one, as they are all equally important. The MS rule, that gives high priority to the tasks with the least amount of slack, is scaled by the look-ahead parameter K_1 . Finally, the SST rule, that discourages long setup times, is scaled by the look-ahead parameter K_2 . Notice that the WSPT is exponentially discounted twice by the MS and SST rules. The reason for this is that the use of an exponential decay function has been found to perform better than a linear decay function [Ow and Morton, 1989, p. 182].

Determining the values of the two look-ahead parameters, K_1 and K_2 , depends on the specific problem instance as they perform a scaling which must be adjusted according to the scale of the processing times, due dates and setup times. As it turns out this is not an easy task and for this reason the determination of optimal values for the look-ahead parameters has been a subject of research, as seen in Chen et al. [2010] and Lee et al. [1997].

In this thesis two different approaches to the determination of the look-ahead parameters will be studied. The first approach, described in the following section, is the determination of the parameters based on the work done by Lee et al. [1997] where the values are calculated based on statistics of the applied task set. This is then followed by a simple trials approach, where the ATCS rule is applied with a range of different values for K_1 and K_2 respectively, followed by an evaluation to identify the best of the calculated solutions.

Determining the look-ahead parameters using statistics

In this approach a set of task specific factors (statistics) are calculated first, and from these the K_1 and K_2 parameters are then calculated. The first factor is the *due date tightness factor*, τ [Pinedo, 2009, p. 446]:

$$\tau = 1 - \frac{\overline{DD}}{C_{\max}} \quad (5.6)$$

Where \overline{DD} is the average due date. If the due dates are loose τ will be close to zero, and conversely a value of one will indicate tight due dates. Next is the *Due Date Range factor*, R [Pinedo, 2009, p. 446]:

$$R = \frac{DD_{\max} - DD_{\min}}{C_{\max}} \quad (5.7)$$

When the difference between the due dates are large the value of R will be high, while a low value will indicate that the due dates are close to each other. Notice that both Equation 5.6 and 5.7 uses the makespan, C_{\max} . As the makespan depends on the scheduled sequence, this needs to be estimated. In this case the following simple approximation is used [Pinedo, 2009, p. 447]:

$$\hat{C}_{\max} = \sum_{j=1}^n p_j + n\bar{s} \quad (5.8)$$

Where n is the number of tasks. This estimate of the makespan can then be used in Equation 5.6 and 5.7 instead of the actual value. The third and final factor to calculate is the *setup severity factor*, η [Pinedo, 2009, p. 447]:

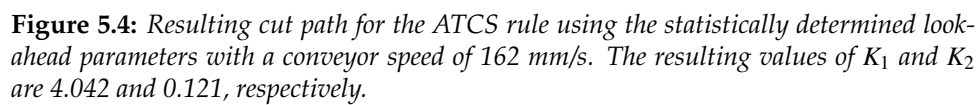
$$\eta = \frac{\bar{s}}{\bar{p}} \quad (5.9)$$

If the setup times are generally small compared to the process times η is small, and conversely it will be large when the setup times are large compared to the process times.

Once the three factors described above has been calculated, the values of K_1 and K_2 can be calculated using the rules [Pinedo, 2009, p. 447]:

$$\begin{aligned} K_1 &= 4.5 + R & \text{for } R \leq 0.5 \\ K_1 &= 6 - 2R & \text{for } R \geq 0.5 \\ K_2 &= \frac{\tau}{2\sqrt{\eta}} \end{aligned} \quad (5.10)$$

By using the above calculations of the look-ahead parameters, the ATCS rule is applied to the scheduling problem, yielding the path shown in Figure 5.4.



However, the path was obtained with a conveyor speed of 162mm/s and as soon as the conveyor speed was raised above this threshold the path was changed to a less feasible path with some late cutting tasks. This is a direct result of the statistic calculation of the look-ahead parameters that changes as the conveyor speed is changed. It also indicates that the look-ahead parameters are not optimally selected for this specific set of tasks. Instead, the path obtained at the conveyor speed of 162mm/s could be used to obtain the conveyor speed of 196mm/s.

5 Dispatching Rule Scheduler

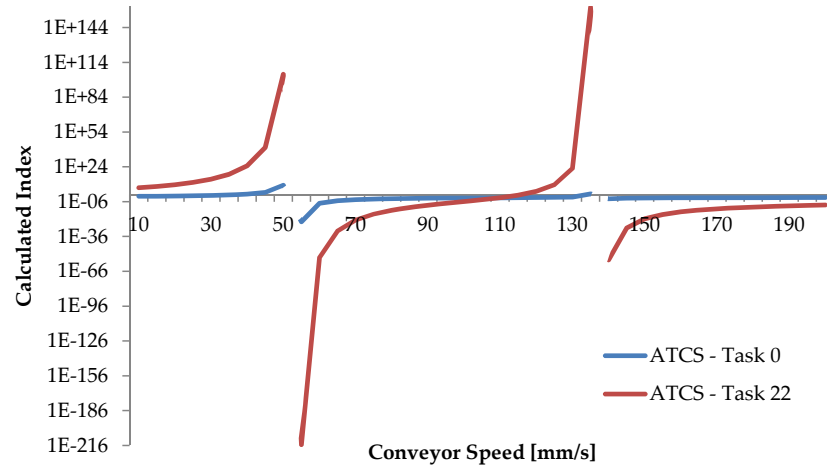


Figure 5.5: The calculated ATCS for tasks 0 and 21 as a function of the conveyor speed. The index is for the first task in the sequence, with fixed cutting- and movement speeds.

By a closer inspection this behavior is result of the use of the exponential discounting of the MS and SST rule, respectively, that was discussed previously. Around a conveyor speed of 135mm/s, the value of the parameter K_2 changes sign inside the exponential SST term (see Equation 5.5), causing a major shift in the index values. This shift is, as it is also seen by Figure 5.5, much more significant for task 21 because this task has a larger setup time than task 0. At a conveyor speed of 50mm/s, the value of the parameter K_1 that changes sign inside the MS exponential term (see Equation 5.5), again causing a major change of the index values. Again, the change is largest for task 21 as this also has much more slack than task 0.

In summary, there are two general issues with this approach to the determination of the look-ahead parameters. First of all, the parameter values does not yield a good solution for the higher conveyor speeds, even though a better solution is clearly found at a lower speed. Secondly, the ATCS calculation does not show the desired behavior for all possible conveyor speeds. For these reasons, the determination of the look-ahead parameters using task specific statistics does not seem feasible. Instead, this motivates the identification of another approach to determining the optimal parameter values.

Determining the look-ahead parameters through trials

Instead of determining the look-ahead parameters K_1 and K_2 from the statistics, this approach determines the best possible values by through trials. Specifically the ATCS rule is applied to the scheduling problem with a complete range of values for both K_1 and K_2 . This gives a large number of possible solutions, one for each possible combination of K_1 and K_2 within their test range. The obtained solutions are then post-processed to check their validity and identify the best possible solution. Hence,

this approach uses the following general steps:

1. *Pre-processing and loading of the scheduling problem.* This includes the loading of data, calculation of processing times and due dates from the cut-, movement- and conveyor speeds. The value ranges of K_1 and K_2 that should be tested is also loaded at this point.
2. *Calculation of the ATCS rule solutions for each combination of K_1 and K_2 .* In this step, the ACTS rule is applied to the scheduling problem for each possible combination of the K_1 and K_2 values within their respective test ranges (loaded in step 1). The resulting solution sequences are then saved for post-processing.
3. *Post-process the obtained solutions and remove invalid solutions.* For each of the obtained sequences found in the previous step, are then checked for validity in terms of satisfying all of the due dates. The sequences that pass this check are then saved to a solution space.
4. *Choose the best obtained solution.* From the remaining sequences (if any) the sequence with the lowest cycle time is chosen and returned as the solution to the scheduling problems.

By applying the steps above to the scheduling problem, the solution space shown in Figure 5.6 is obtained for test ranges $K_1 = [0.1, 0.2, \dots, 10]$ and $K_2 = [0.1, 0.2, \dots, 3]$ yielding a total of 3000 solutions. These ranges have been selected based on some initial trial runs, and may be necessary to change for other task sets than the current scheduling problem. Identifying the smallest cycle time from the solution space in

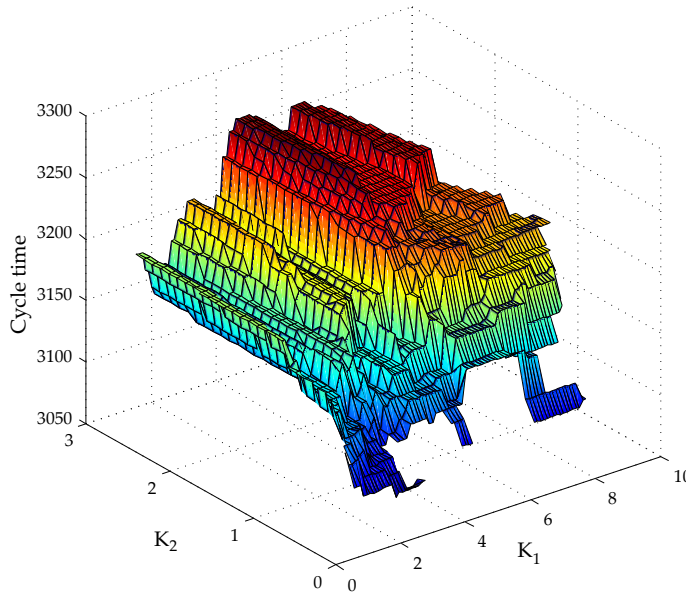


Figure 5.6: The resulting cycle time of the ATCS algorithm as a function of K_1 and K_2 . Only the valid solutions where all due dates are satisfied are shown.

Figure 5.6 yields the parameters $K_1 = 2.5$ and $K_2 = 0.2$ and the solution path shown in Figure 5.7. The obtained path has a cycle time of 3081 ms equivalent to a conveyor

Travel distance: 1695 mm

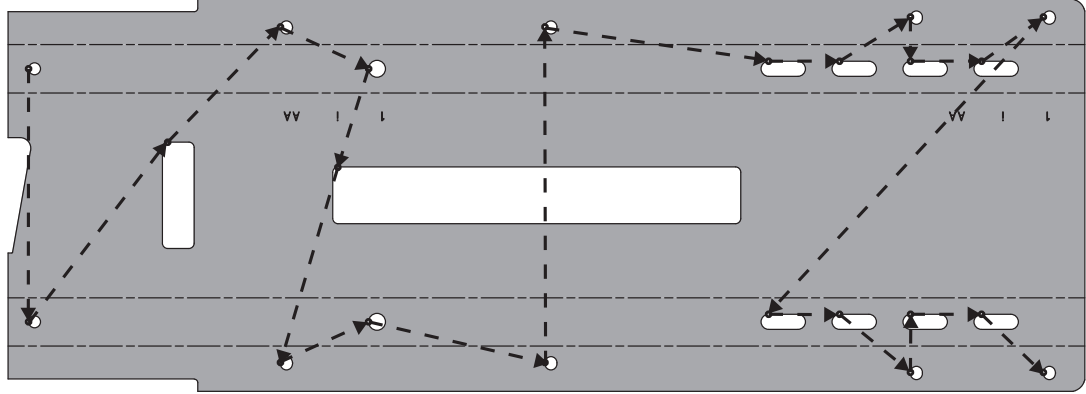


Figure 5.7: Resulting cut path for the ATCS rule using the trials approach with a conveyor speed of 197 mm/s. The resulting values of K_1 and K_2 are 2.5 and 0.2 respectively.

speed of 197 mm/s. Comparing the path with the one shown in Figure 5.4 it is observed that the two paths are very similar and are only deviate from each other in the last part of the paths. Even so, the path obtained is slightly shorter and hence it also has a lower cycle time.

Recalling the problems that was observed with the statistic determination of the K_1 and K_2 in the previous section, these are all avoided in this approach as their values are selected strictly in terms of their performance. The tradeoff for using this approach is the additional required computational time when compared to both the approach using statistics, as well as the basic dispatching rules. In this case however this is not an issue as the scheduling will not be required to be carried out online. At the same time the calculation is still fairly quick, as the calculation time of the solution above is around 1.5 seconds on a Intel Core™2 Duo Processor T7200 @ 2Ghz with one thread. It should however be kept in mind that the calculation time depends on the ranges and value intervals of the K_1 and K_2 parameters.

Based on the above, this approach offers a stable and fairly quick solution to the scheduling problem with the objectives of minimizing the makespan and the maximum lateness. It is however necessary to emphasize that the solution is not necessarily optimal, even though it might offer a near optimal solution.

5.3 Summary

This chapter has investigated the composition and use of a dispatching rule scheduler to solve the scheduling problem that was described in Section 4.2.

The use of basic dispatching rules was investigated first, and although some valid solutions were obtained they were not good as a direct result of their inability to satisfy multiple objectives. The MS rule introduced some dynamic behavior into the scheduling problem in satisfying the due dates, but yielded a long path. Contrary to this the SST rule produced a shorter path, but was not able to meet the due dates.

To combine the best of the MS and the SST rule, the composite ATCS dispatching rule was studied next. To do this the ATCS rule introduced the two look-ahead parameters K_1 and K_2 that determined the contribution from the MS and SST rule, respectively. To determine the value of these parameters two approaches were studied; using statistics and trials. The latter approach proved to be the most stable approach while at the same time yielding the best overall solution for the scheduling problem considered in this thesis. The tradeoff for the approach was an additional, however insignificant, cost in computational time.

CHAPTER 6

Combinatorial Optimization

In contrast to scheduling with dispatching rules the problem of scheduling can also be considered to be a problem of combinatorial optimization. Combinatorial optimization is a topic that basically consists of finding an optimal solution from a finite set of solutions [Schrijver, 2003, p. 1].

This chapter focuses contrary to Chapter 5 on finding such an optimal solution to the laser cutting problem. Section 6.1 focuses on running times for algorithms and argues that a simple enumeration of all possible $(n - 1)!$ permutations and looking the best solution is not feasible. Section 6.2 describes the Travelling Salesman Problem, which is a well known problem within the field of combinatorial optimization. The Travelling Salesman Problem resembles the laser cutting problem and the methods of solving this problem is therefore studied. Finally Section 6.2 sums up what solution method will be used to tackle the laser cutting problem. Finally this chapter discusses the software used and the implementation.

6.1 Running Time of Algorithms

A simple approach to solving the laser cutting problem is enumerate all possible solutions and search for the best one. Finding the optimum solution this way is also known as brute force search. An assessment of whether this approach is feasible is needed.

If all permutations are calculated it will take at least $(n - 1)!$ steps. This leads to a calculation time that is proportional to $(n - 1)!$ which is clearly impractical. Putting the direct solution of instances with 50 or so points is well out of reach of the combined computing power of all the worlds' machinery [Applegate et al., 2006, pp. 45-46]. This makes it clear that *polynomial time algorithms* are preferred, also called *good algorithms* [Edmonds, 1965]. Polynomial time algorithms has a computation time proportional to $O(n^c)$ using the "Bog O Notation".

Table 6.1 shows the maximum input sizes solvable within one hour with different types of algorithms. In (a) one elementary step is assumed to take one nanosecond while (b) assumes a ten times faster machine.

	$10n^2$	$n^3 .5$	2^n	$n!$
Case (a)	60,000	3,868	41	15
Case (b)	189,737	7,468	45	16

Table 6.1: Maximum input size solvable within one hour [Korte and Vygen, 2002, p. 7, Table 1.2].

The table clearly shows the advantage of polynomial-time algorithms. They can handle larger instances in reasonable time at the size of the solvable instances increases considerably with machine power. The opposite is the case for exponential-time algorithms.

The above shows that depending on the problem size focus is needed on the concepts of polynomial-time algorithms and exponential-time algorithms. There are problems where no polynomial-time algorithm exists, and there are problems for which there exist no algorithms at all. The laser cutting problem described in subsection 4.3 is equivalent to the famous “Travelling Salesman Problem” (TSP) and is known not to have an exact polynomial-time algorithm. However it is possible for many problems of this kind to find approximate solutions in polynomial time. The next section will look into how the TSP can be solved without using a brute force search.

6.2 The Travelling Salesman Problem

The Travelling Salesman Problem (TSP) is an NP-hard problem in combinatorial optimization. The problem is almost identical to the laser cutting problem stated mathematically in subsection 4.3 and it is thus of great importance when solving the laser cutting problem. The definition of the TSP is simple: Given a list of cities and their pair wise distances, the task is to find a shortest possible path that visits each city exactly once and returning to the starting point [Applegate et al., 2006, p. 1]. Figure 6.1 shows an example of a 33-city TSP instance and solution. In fact the only difference between the laser cutting problem and the TSP is, that the TSP tries to find a *Hamiltonian cycle*¹ in a graph with minimum cost where the laser cutting problem tries to find a *Hamiltonian path*² in a graph with minimum cost.

¹A Hamiltonian cycle (or Hamiltonian circuit) is a cycle in an undirected graph which visits each vertex exactly once and also returns to the starting vertex.

²A Hamiltonian path (or traceable path) is a path in an undirected graph that visits each vertex exactly once.

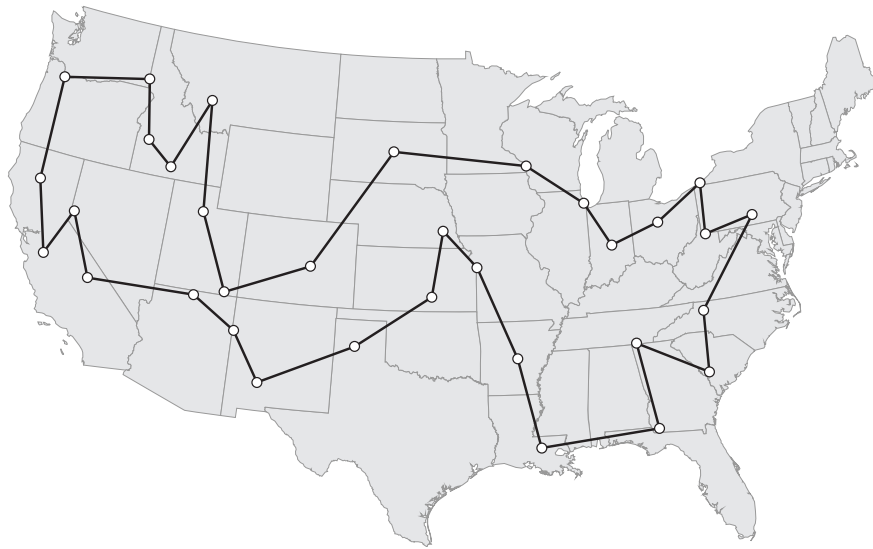


Figure 6.1: Optimal 33-city path in USA [Applegate et al., 2006, p. 15, Figure 1.10].

The problem was formulated as a mathematical problem in 1930 and is without a doubt one of the most extensively studied problems in optimization [Korte and Vygen, 2002, p. 473]. The TSP has several applications in real life and is used as benchmark for many optimization methods.

Because the TSP and the laser cutting problem are similar, it is useful to study how to solve the TSP and then evaluate how, and to what extent, these solution methods can be applied to the laser cutting problem. The solution methods are classified in Section 6.2.1 after which the most important methods will be commented.

6.2.1 Classification of Solution Methods

When solving combinatorial optimization problems like the TSP it is very important to distinguish between heuristics and algorithms. Unfortunately the use of these terms is often used randomly in the literature. The definition used in this thesis is as follows:

An algorithm is a list of well-defined instructions for calculating a function. The simplest is the exhaustive search that enumerates all possible solutions and subsequently picks the best one.

A heuristic is an experience-based technique for problem solving. They speed up the process of finding a solution, where an exhaustive search is impractical. However there is no guarantee of finding an optimal solution or even a solution of good quality.

This definition is not very precise as the difference between heuristics and algorithms can be discussed. Hopefully the difference when solving the TSP is obvious to the

reader.

A selection of heuristics and algorithms is presented in Figure 6.2.

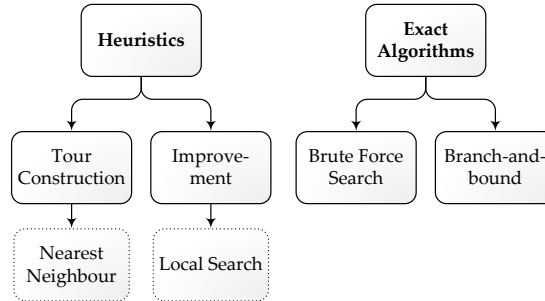


Figure 6.2: Important heuristics and algorithms for solving the TSP, inspired by [Nilsson, 2003].

A path construction heuristic is often used for generating an initial path that is used as input for an improvement heuristic. An example of a path construction heuristic and a path improvement heuristic is presented next along with an introduction to exact algorithms.

6.2.2 Path Construction

One of the first heuristics to determine a solution to the TSP is the nearest neighbour algorithm. It is easy to implement and executes quickly, but rarely yields good solutions. An implementation has been made in MATLAB that illustrates one of the problems with this approach, see Figure 6.3. The problem is, that the algorithm can sometimes cause you to "paint yourself into a corner", requiring long edges to get back to the unvisited cities [Applegate et al., 2006, p. 104]. Figure 6.3 shows this issue with the point in the lower right corner.

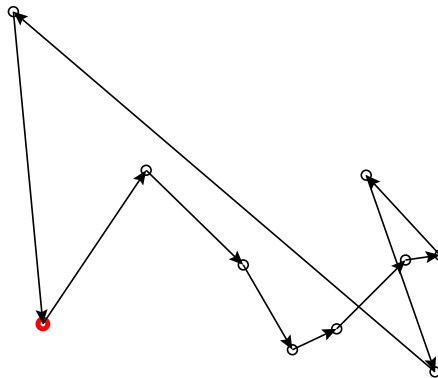


Figure 6.3: Nearest Neighbour search using MATLAB. The red dot marks the starting point.

The nearest neighbour search algorithm is only of interest because of its simplicity.

The complexity of the nearest neighbour search is $O(n^2)$. By considering all vertices as a possible starting point the complexity is $O(n^3)$ [Laporte, 1992, p. 242]. This heuristic has been analysed in terms of performance and it is guaranteed to perform within these bounds [Rosenkrantz et al., 1977, p. 565]:

$$\text{Worst Case Performance: } \frac{\text{Nearest Neighbour Solution}}{\text{Optimal Solution}} \leq \frac{1}{2} \cdot \left\lceil \frac{\log(n)}{\log(2)} \right\rceil + \frac{1}{2} \quad (6.1)$$

with n denoting the problem size (number of cities) and $\lceil x \rceil$ denotes the smallest integer greater than or equal to x .

Figure 6.3 constitutes a travelling salesman graph with ten nodes. According to Equation 6.1 the length of the nearest neighbour solution is less than 250% of the optimal solution length. This example clearly shows that the performance of the nearest neighbour heuristic is far from optimal and cannot be used without a path improvement heuristic in situations where the path length is critical.

Many other construction heuristics are available e.g. the insertion heuristic and the patching heuristic [Laporte, 1992, pp. 242-243]. In fact the dispatching rules presented in Chapter 5 can also be considered to be construction heuristics. The dispatching rule Shortest Setup Time First (SST) corresponds to the nearest neighbour heuristic when the setup times are considered proportional to the distance between the tasks. The performance of the SST rule can in this case also be evaluated with Equation 6.1.

6.2.3 Path Improvement

Path improvement is often carried out using local search heuristics. The input to these heuristics is an approximate solution made from one of the construction heuristics. Then the heuristic systematically improves the solution by certain “local” modifications [Korte and Vygen, 2002, p. 485]. The basic structure of the heuristic is to replace pairs of path edges by cheaper alternative pairs where available. The idea of such an operation is shown in Figure 6.4.

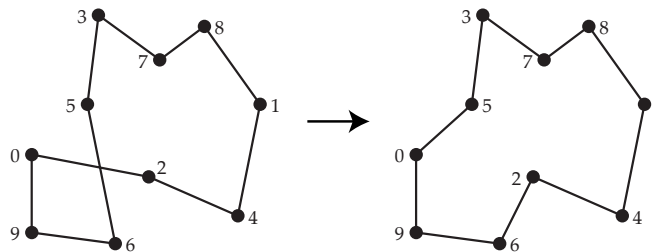


Figure 6.4: A local modification [Merz and Freisleben, 2001, p. 301].

This algorithm has shown to be effective and works well for instances with up to

100 points. It is simple to implement so it should be considered a possibility for the laser cutting problem.

These local search methods were first introduced by Lin and Kernighan in 1973 and they made a great improvement in the quality of paths provided by heuristic methods. Even today these methods remain a key ingredient in many approaches for finding quality paths [Applegate et al., 2006, p.104].

6.2.4 Exact Algorithms

In practice the brute force search is often carried out using a *depth first search* (DFS) or *breadth first search* (BFS). Both methods are merely a way of looking through all nodes in a search tree, see Figure 6.5.

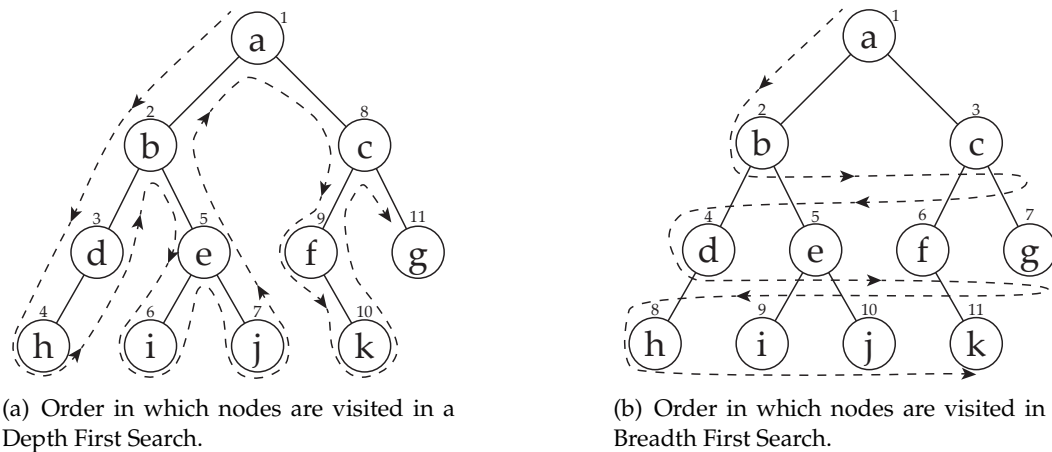


Figure 6.5: Methods of exhaustive search [Stefik, 1995, pp. 165-167]. Each node in the search trees represents a feasible solution to the TSP. The numbers indicate the order in which the nodes are visited.

In DFS the processor descends through the search tree by going deeper and deeper until a node has no children. Then the search backtracks, returning to the most recent node that has unvisited children.

The BFS searches through all nodes at a given level before checking children. Comparing the performance of DFS and BFS is difficult as it depends very much on the problem type and the search tree. However when the search tree is very large neither approach is particularly good [Stefik, 1995, p. 170]. When the search tree is large it is important to have guidance about which direction to go and Branch-and-Bound is an example of how this can be done.

The branch-and-bound technique was invented in the late 1950s [Applegate et al., 2006, p. 94]. It is used in a wide range of optimization problems. The technique

basically consist of enumerating all solutions and discarding large subsets of solution candidates by using upper and lower bounds of the objective function.

Let us consider a TSP problem. A path can be represented as its incidence vector x of length $n(n-1)/2$, with each component of the vector set at 1 if the corresponding edge is a part of the path, and at 0 otherwise [Applegate et al., 2006, p. 83]. The distances between cities are stored in a vector c . The vectors c and x are indexed by the pairs of cities. Thus $c^T x$ gives the cost of the path. Letting Ψ denote the set of the incidence vectors of all paths, the problem is to:

$$\text{minimize } c^T x \text{ subject to } x \in \Psi \quad (6.2)$$

The first tool in the branch-and-bound technique is a splitting procedure. The point is to divide a given set of solution candidates into two or more smaller subsets. In this case a vector α and numbers β', β'' with $\beta' < \beta''$ can be chosen such that each $x \in \Psi$ satisfies either $\alpha^T x \leq \beta'$ or $\alpha^T x \geq \beta''$. The problem can now be divided into two sub problems:

Thus there are two sub problems:

$$\text{minimize } c^T x \text{ subject to } x \in \Psi \text{ and } \alpha^T x \leq \beta' \quad (6.3)$$

and

$$\text{minimize } c^T x \text{ subject to } x \in \Psi \text{ and } \alpha^T x \geq \beta'' \quad (6.4)$$

This specific step is called branching. This is easy to remember when thinking about how this step relates to the *branch-and-bound tree*. A branch-and-bound tree is presented in the next section along with an example of solving the TSP using branch-and-bound. The sub problems correspond to the nodes in the branch-and-bound tree where the original problem is the root of the tree and each sub problem leads to a branching step.

After branching a procedure computes a upper and lower bound for the minimum value of $c^T x$ within the given subset of Ψ . This step is called *bounding*. The idea behind branch-and-bound is to discard (called *prune*) nodes where the lower bound is greater than the upper bound for some other node. Usually a global variable keeps track of the minimum upper bound seen among the sub regions examined so far [Applegate et al., 2006, pp. 94-95].

The branch and bound algorithm stop in two cases: When the set Ψ has been reduced to a single element or when a solution is found that matches the lower bound. Either way the branch-and-bound method will find the optimum solution. An example of how the branch and bound algorithm can be used to solve an actual

TSP is shown next.

6.2.5 Example of Solving TSP Using Branch and Bound Algorithm

This section goes through an example that shows how to solve a five city TSP problem using branch and bound. The reader should be aware that the branch and bound method is not one well defined method and can be carried out in a number of ways. The method used here was developed by Eastman in the late 1950s and early 1960s [Lawler and Wood, 1966, p. 707]. The branch-and-bound tree for this example can be seen in Figure 6.7.

The approach used by Eastman is to solve the assignment problem corresponding to the TSP. The TSP problem is the same as the assignment problem with the added constraint that the solution must be cyclic, i.e. no sub paths are allowed. If the optimal solution to the assignment problem is cyclic then the solution for assignment problem is the optimal solution for the corresponding TSP problem. If the solution is not cyclic constraints are added until a cyclic solution is achieved.

Let us look at an example. Given the distance matrix in Table 6.2 the corresponding assignment problem is solved. The assignment problem can be solved in $O(n^3)$ time [Korte and Vygen, 2002, p. 236]. For a detailed description of how the assignment problem can be solved, the reader is referred to Korte and Vygen [2002]. Solving the assignment problem yields the solution shown in Figure 6.6. Notice that this solution has two sub-paths and is thus *not* a feasible solution to the TSP.

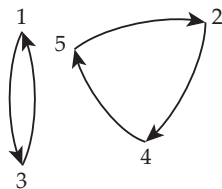


Figure 6.6: Solution to assignment problem with distance matrix given in Table 6.2. The cost is 33.

	$\cdot \rightarrow 1$	$\cdot \rightarrow 2$	$\cdot \rightarrow 3$	$\cdot \rightarrow 4$	$\cdot \rightarrow 5$
$1 \rightarrow \cdot$	-	10	8	9	7
$2 \rightarrow \cdot$	10	-	10	5	6
$3 \rightarrow \cdot$	8	10	-	8	9
$4 \rightarrow \cdot$	9	5	8	-	6
$5 \rightarrow \cdot$	7	6	9	6	-

Table 6.2: Example of distance matrix for five city problem.

The optimum solution to the assignment problem is 33 and this acts as a lower bound (LB) for the TSP. Now there is two sub-paths in the solution to the assignment problem, one being $1 - 3 - 1$ and the other being $2 - 4 - 5 - 2$. In order to eliminate these sub-paths we take the smallest sub-path and create two branches: one with $x_{1-3} = 0$ and one with $x_{1-3} = 1$. This is a way of saying that either x_{1-3} lies in the solution or it does not. x_{1-3} denotes the line from one to three in Figure 6.6. This gives us two new problem instances that can be solved like an assignment problem. The branch and bound search tree is shown in Figure 6.7.

When the assignment problem is solved for $x_{1-3} = 0$ the cost is 34. Thus the lower bound for this node is 34 and there is still two sub-paths. If $x_{1-3} = 1$ the lower bound is also 34 with two sub-paths. In fact the solutions for $x_{1-3} = 0$ and $x_{1-3} = 1$ are the same.

When the assignment problem with the constraints $x_{1-3} = 0$ and $x_{2-4} = 0$ is solved, it gives a feasible solution to the TSP with a value of 36. This value will then act as an upper bound. All nodes with a lower bound more than 36 can now be pruned. When we move on and solve the assignment problem with the constraints $x_{1-3} = 0$ and $x_{2-4} = 1$ we also get a feasible solution to the TSP but this time with a value of 34. As all remaining nodes are equal to or more than this new upper bound the search has finished. This is because that branching from the node with the constraint $x_{1-3} = 1$ will only yield solution that are equal to or more than 34.

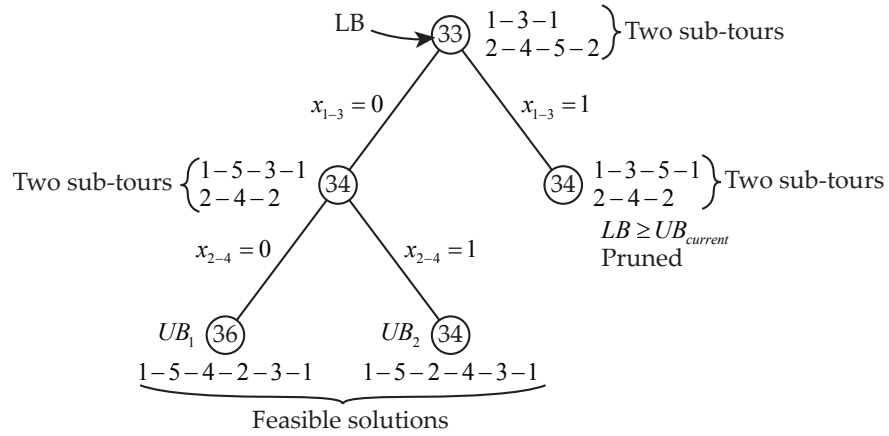


Figure 6.7: Branch-and-bound tree.

6.2.6 Summary

Various solution methods for the TSP has been studied. The branch-and-bound methods is considered the best method for finding optimum solutions. The rest of this chapter will focus on solving the laser cutting problem by applying the branch-and-bound method. Section 6.3 focuses on choosing the right programming framework and software, Section 6.4 describes the implementation and Section 6.5 presents results.

6.3 Software

This section will discuss what method of programming and software package are used for implementing the solver. Firstly the programming methodology will be discussed and afterwards a software package will be chosen.

The laser cutting problem is in fact a constraint satisfaction problem and therefore it is natural to focus on constraint programming. These three approaches are considered:

- ◊ Programming a solver from scratch.
- ◊ Programming using a constraint logic language.
- ◊ Constraint programming via a separate library.

Programming a solver from scratch would be possible but the further implementation of constraints would be impractical and cumbersome. Therefore this possibility is discarded even though it has some pretty obvious advantages by not being tied to a specific programming language.

Constraint logic programming (CLP) is an embedding of constraints in a host language. Historically CLP emerged as a generalization of logic programming [Rossi et al., 2006, p. 411]. CLP provides great power for modelling problems, specifying search heuristics, experimenting with constraint solving techniques and so on. However, CLP languages have significant drawbacks. The features of logic programming must be learnt and understood before it is possible to model and write solvers. There also exists some distinctive programming drawbacks [Rossi et al., 2006, p. 445]. The difficulty of programming in CLP languages was actually identified as a problem in [Sibbald et al., 1992]. Examples of some popular constraint logic languages are B-Prolog, ECLiPSe, SICStus, Oz, GNU Prolog and Turtle.

The most common way of using constraint programming is probably using a separate library. There are a lot of libraries available for free written in Java, C, C++ and Python. Open source libraries are preferred in this project as they allow modification of existing solvers. This rules out the solvers like the ILOG Solver which has been the undisputed leader in performance from the mid 1990s an up to about 2005 [Kotthoff, 2010, p. 2]. The ILOG Solver still performs well but recent developments such as Gecode and Minion competes well with ILOG in terms of performance and memory usage.

Currently Choco, ECLiPSe, Gecode and Minion are under active development and they are all open source. They are all good choices for solving constraint satisfaction problems efficiently. Minion requires only an input file to run and no written code. This way the solver is made fast by not being extensible or programmable. However it would be a concern in this project as it makes the solver less flexible.

Gecode is selected for the further development of a solver to the laser cutting problem. Gecode is chosen partly because it has showed good benchmark results but also because it reportedly should have a large community. Benchmarking has been carried out by Kotthoff [2010]. CLP was discarded because of the mentioned drawbacks.

For an in depth documentation of Gecode the reader is referred to [Schulte et al., 2011].

6.4 Implementation

This section describes the implementation of the solver used to find the optimal solution to the laser cutting problem. The section explains what methods are used to implement constraints but does not present a detailed guide to understanding the source code which is found in Appendix C.

Remember that the implementation is based on the mathematical description given in Section 4.3. This means that the solver finds a permutation of the points shown in Figure 4.3 where each point corresponds to the start/stop point for a task. According to the mathematical definition of the laser cutting problem the process time is reduced by reducing the distance travelled when not cutting.

The following sections explain the implementation step by step.

6.4.1 Circuit Constraint

Gecode supports only one graph constraint: The circuit constraint. The solver is therefore based on this constraint.

The circuit constraint forms a Hamiltonian circuit which is a cycle in a graph that visits each vertex exactly once and also returns to the starting vertex [Rossi et al., 2006, pp. 180-181]. The circuit constraint is implemented by calling `circuit(home, c, x, y, z)`; where `c` is a cost or distance matrix, `x` defines the values forming the circuit, `y` defines the cost of the edge for each node, `z` defines the total cost of the edges in the circuit.

When using this constraint all that needs to be defined is the cost matrix `c`. The process is shown in Equation 6.5 where the circuit constraint outputs a permutation given the calculated distances as input.

$$\underbrace{\begin{pmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ \vdots & & \\ x_n & y_n & z_n \end{pmatrix}}_{\text{Loaded vertices}} \rightarrow \underbrace{\begin{pmatrix} d_{11} & d_{12} & \cdots & d_{1n} \\ d_{21} & d_{22} & \cdots & d_{2n} \\ \vdots & & & \\ d_{n1} & d_{n2} & \cdots & d_{nn} \end{pmatrix}}_{\text{Calculated distances}} \rightarrow \underbrace{\{1, \dots, 1\}}_{\text{path}} \quad (6.5)$$

The circuit constraint only constructs the Hamiltonian cycle and is thus combined with a minimize script that searches for the lowest value of `z`.

6.4.2 Hamiltonian Path

In Gecode there is a number of ways to implement variable constraints. It can be done intuitive using expressions like this example: `rel(home, z == 3*x-4*y+2);`. This section presents constraints that are implemented using such expressions.

Recalling that a solution to the laser cutting problem is a Hamiltonian path a constraint is added that forces the vertex n to be before the final vertex 1. The constraint relates to x in the circuit constraint that defines the values forming the circuit.

The circuit constraint outputs the total cost of the path as z . z contains the cost of the Hamiltonian cycle. A constraint is added that calculates a value, z_{path} , that is the cost of the Hamiltonian path instead:

$$z_{\text{path}} = z - d_{n1} \quad (6.6)$$

Equation 6.6 calculates the cost of the Hamiltonian path by subtracting the distance from the final point to the start point from the cost of the Hamiltonian cycle. By minimizing z_{path} the output of the solver is the optimum Hamiltonian path between point 1 and n .

Due dates and release date are calculated using the formulas described in Section 4.4.2. Using the calculated due dates and release dates the following constraints are applied:

$$C_j < DD_j \quad (6.7)$$

$$C_j - P_j > RD_j \quad (6.8)$$

P_j is subtracted from C_j in the constraint regarding the release date because the entire task needs to be inside the cutting area *before* the processing of the task may be started.

6.4.3 Input and Output

The coordinates of all vertices are given as input through the text file "coordinates.txt". The columns in the text files are x_{\min} , x_{\max} , y , z and cut length. The cut length refers to how many millimetres to cut. This is used to calculate the process time for the task.

In the file "constants.txt" the conveyor speed v_{conveyor} , cut speed v_{cut} and move speed v_{move} are specified. The move speed refers to the speed at which the laser beam can be moved when not cutting. The numbers are all in mm/s.

The output of the solver is written to "result.txt". The file contains the final path and the total distance. To improve system testing the solver will also output extended

results to "resultextended.txt" which contains start and finish time for all geometries along with due and release dates. Furthermore this text file contains all feasible solutions and not just the best. An example of the two text files are given in Table 6.3 and 6.4 (where just one solution is found):

	Point	Start	Finish	DD	RD
0	0	0	324	10580	-19080
2	2	408	2695	14350	-14750
1	1	2746	3071	10580	-19080
4	4	3102	3448	17680	-11980
7	7	3507	3961	20165	-9360
5	5	3992	11855	19150	650
3	3	11927	12273	17680	-11980
6	6	12323	12777	20165	-9360
8	8	12827	13173	25130	-4530
9	9	13284	13630	25130	-4530
Cost: 922		Cost: 922			

Table 6.3: Example of "result.txt".

Table 6.4: Example of "resultextended.txt". All figures are given in milliseconds.

A sample of the file "coordinates.txt" is shown in Table 6.5. Note that the terms x_{min} and x_{max} are defined as described in Section 4.4.2. The file "constants.txt" contains three lines with the cutting-, move- and conveyor speeds in mm/s, respectively.

x_{min}	x_{max}	y	z	Cut Length
11.6	18.4	-16.2	0.0	21.4
11.6	18.4	-158.9	0.0	21.4
87.0	105.0	-60.5	0.0	150.9
153.6	160.4	6.8	0.0	22.8

Table 6.5: Example of "coordinates.txt". All figures are given in millimeters. Only the numbers written in *verbatim* is included in the file.

6.5 Results

This section will present the most significant results. The following implementations will be discussed:

- ◇ Clean TSP (Hamiltonian Cycle)
 - Analyse calculation time.
- ◇ Hamiltonian path

- Evaluate output with different end points.
- ◇ Hamiltonian path with due dates and release dates
 - Evaluate with cutting and move speed of CO₂ laser cutter.
 - Evaluate with cutting and move speed of remote laser cutter.

The clean TSP solver will be used to analyse calculation time as a function of problem size.

The results from the Hamiltonian path solver will show how the solver finds the optimum path based on the defined start and end point. The presentation of the implementation with due dates and release dates will show how the resulting path is affected by the cutting, movement and conveyor speed. This will be done with the process characteristics of a CO₂ laser cutter and a remote laser cutter. The purpose of this section is to demonstrate the capabilities of the solver and not necessarily to evaluate different production scenarios.

6.5.1 Clean TSP (Hamiltonian Cycle)

The TSP is solved using Branch and Bound. The solver will be analysed with respect to calculation time. This gives an indication of how the calculation time is dependent on the problem size. The result is shown in Figure 6.8.

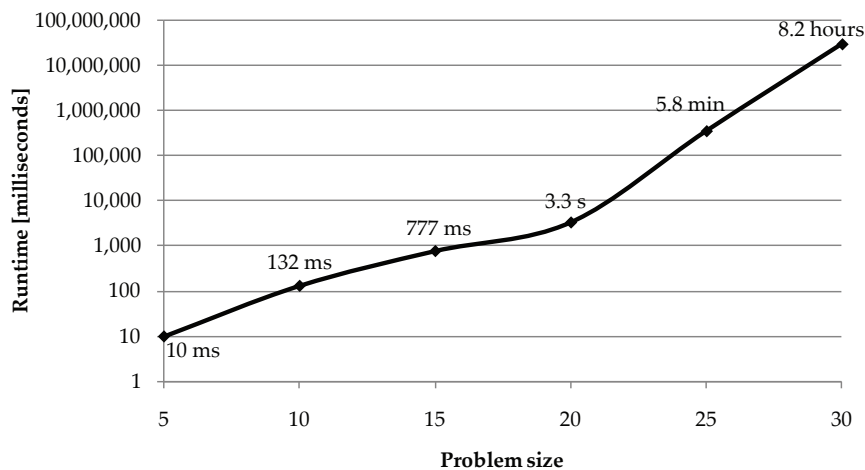


Figure 6.8: Runtime of clean TSP solver on Intel Core™2 Duo Processor E8400 @ 3.00 GHz with one thread.

Figure 6.8 shows that the runtime increases exponentially in terms of problem size (note that the y-axis is logarithmic). It is difficult to solve problems with more than 30 points. It is important to note that it would be possible to improve the runtime by using more threads with a multi core processor. Even though the use of more threads

would increase the computing power, it would only provide a small improvement in terms of how big problems can be solved in a given time. This was also shown in Table 6.1.

6.5.2 Hamiltonian Path

When generating a Hamiltonian path, the solver does not allow for an undefined end point. How the selection of end point affects the result of the solver is therefore evaluated. The geometry presented in Figure 4.2 is used to illustrate the results of the solver.

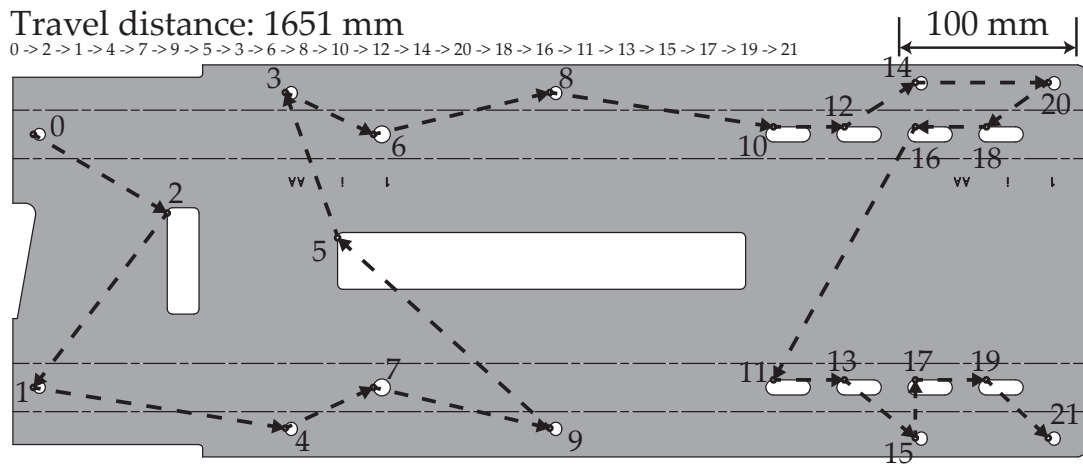
Figure 6.9 shows different paths based on different end points. In the three cases the same start point is used. It is interesting to observe how the path and travel distance changes depending on the choice of end point.

Travel distance does not vary much between the three paths. The shortest path is in Figure 6.9(c) where the path is allowed to follow the edge of the part. When the starting and end point is in separate ends of the part, it is necessary for the path to cross the part several times which gives a longer path.

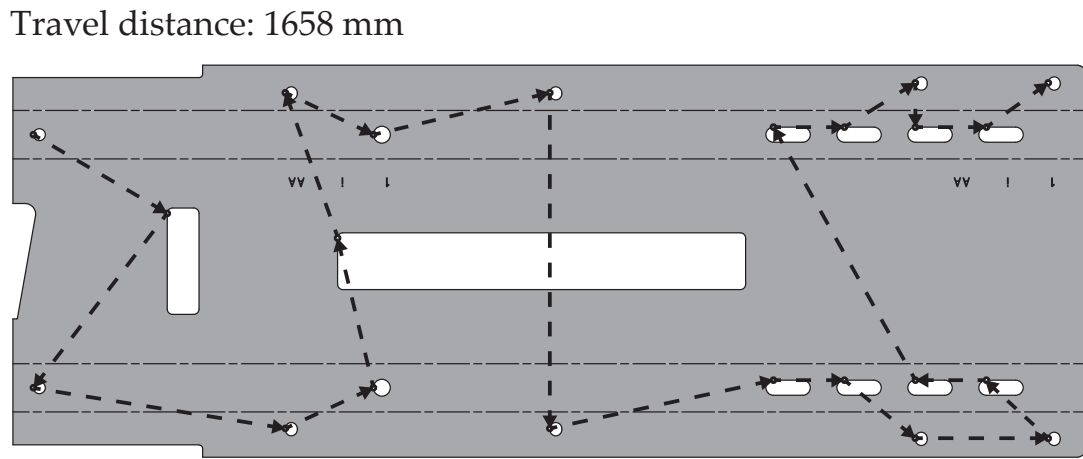
It is possible to calculate all paths within ten seconds³, which is an acceptable calculation time. These calculation times also agree well with the calculation times for solving the TSP given in Figure 6.8.

The results in Figure 6.9 shows that the defined end point does not have a big influence on the total travel distance.

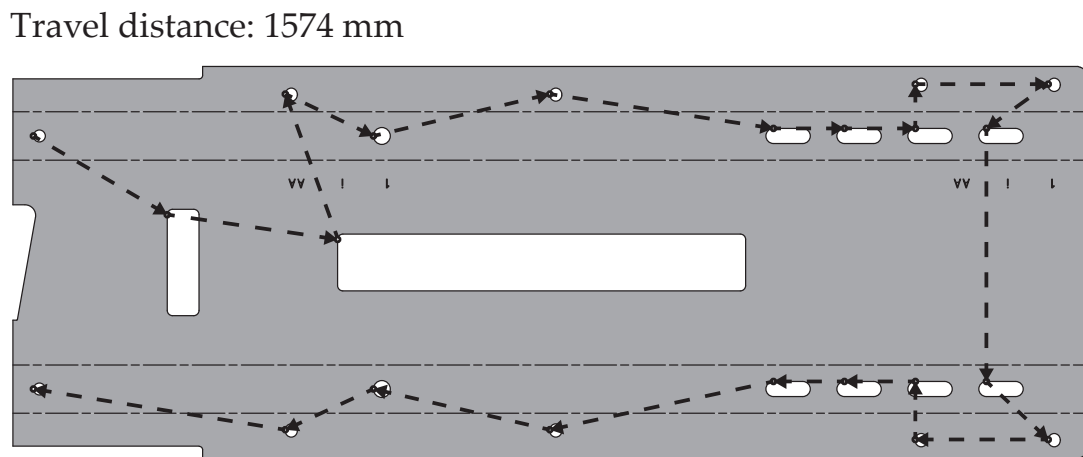
³Using a Intel Core™2 Processor T5600 @ 1.83 GHz.



(a) Illustration of how the points on the part are enumerated and the calculated path with end point 21.



(b) Path with end point 20.



(c) Path with end point 1.

Figure 6.9: Calculated path with predefined end point.

6.5.3 Hamiltonian Path with Due Dates and Release Dates

Before executing this solver, the process data must be specified. This is because the movement of the part is now considered. The parameters presented in Table 6.6 is used as input. The cutting speed and movement speed are based on information from [The Danish National Advanced Technology Foundation, 2010] as well as a data sheet on Kuka Systems GmbH [2011].

CO₂	
Cutting Speed	4 m/min = 67 mm/s
Movement speed	100 m/min = 1667 mm/s
Cutting area	600 mm
Delay	200 mm
Remote laser cutting	
Cutting speed	30 m/min = 500 mm/s
Movement speed	470 m/min = 7833 mm/s
Cutting area	600 mm
Delay	200 mm

Table 6.6: Process parameters used for testing solver.

The cutting areas are based on estimates and can change for any given production scenario. In this case the cutting area is almost the same as the length of the part.

Another process characteristic that needs to be specified is at what time the laser cutter begins cutting. The laser cutter could begin immediately after the first task is fully within the cutting area. However this may not be realistic since the laser cutter may have a job to complete before. Therefore a "delay" is introduced. A delay of 200 mm means that the part is 200 mm from the end of the cutting area as the laser cutting starts. This is illustrated in Figure 6.10.

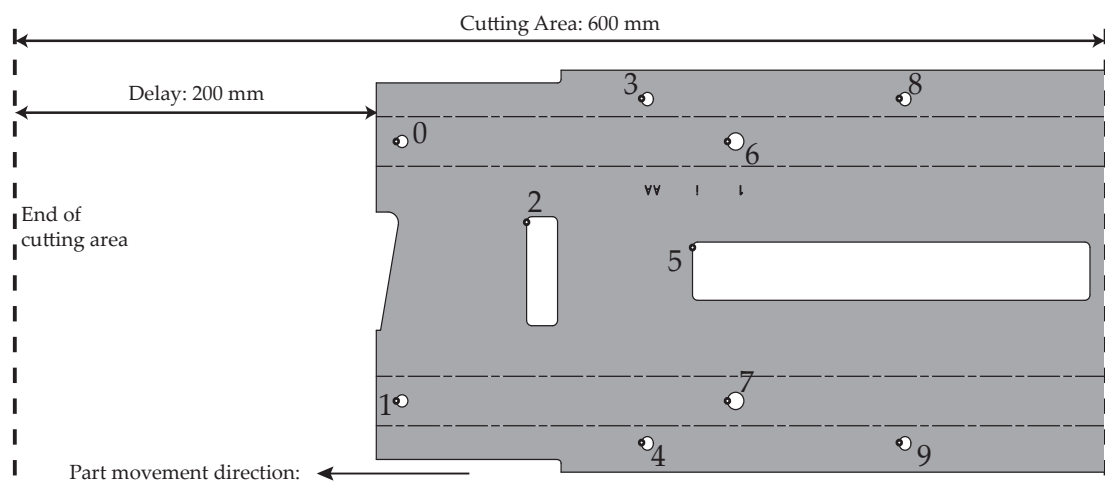


Figure 6.10: Illustration of delay in relation to the part and cutting area.

The delay is important to ensure that the solver will find solutions for the problem when using due dates and release dates. It is however important to look at the delay to ensure that a scheduling result can be used for continuous cutting. This is discussed later.

Next the results from the solver are presented. First the process characteristics of a CO₂ laser is used.

Scheduling for CO₂ Laser Cutter

The start and end point seen in Figure 6.9(a) is used. Figure 6.11 shows a set of process settings that yields a feasible solution.

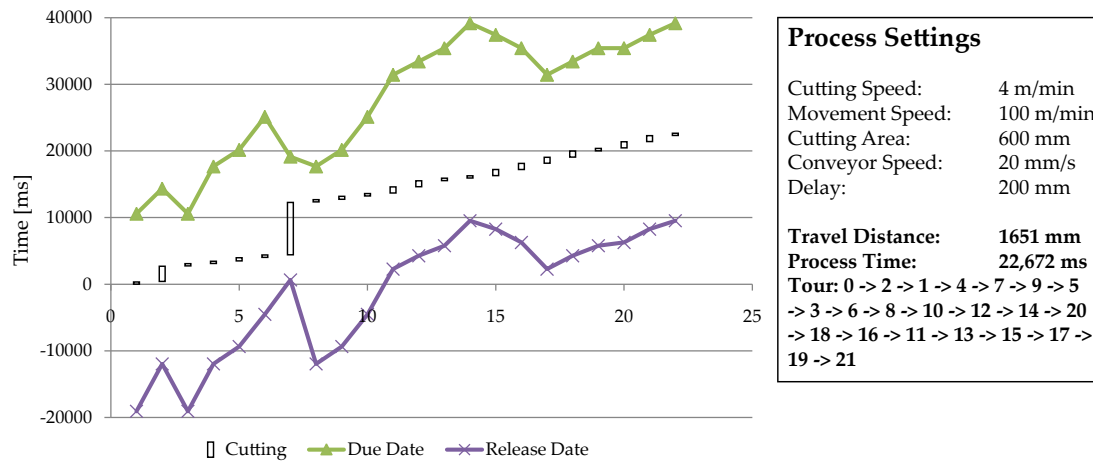


Figure 6.11: Chart showing scheduling results with CO₂ laser cutting process data.

Figure 6.11 shows due and release dates for each task as well as a column showing start and finish time for each task. The total travel distance for this solution is 1651 mm. From this it is concluded that the due dates and release dates are not influencing the path as the path is optimal. It is shown in Figure 6.9(a) that the optimum path length to this problem is 1651 mm. From the scheduling solution a total process time will be 22.7 s is estimated.

The process settings affects the due dates and release dates. As seen in Figure 6.11, the due and release dates can be represented as lines. The process settings are listed below along with a description of how they affect the due date and release date lines. The only process settings that does not affect the due and release dates are the movement and cutting speed.

Cutting Area affects the release date. An increase in cutting area will shift the release date line downwards.

Delay shifts both the due date and release date line by $\Delta\text{delay}/v_{\text{conveyor}}$.

Conveyor Speed has a bit more complex effect. With respect to the due date line an increase in conveyor speed will lower the line and the slope of the line. An increase in the conveyor speed will essentially have the same effect on the release date line. Note however that the negative release dates will be less negative as the conveyor speed is increased, see an example of this in Figure 6.12.

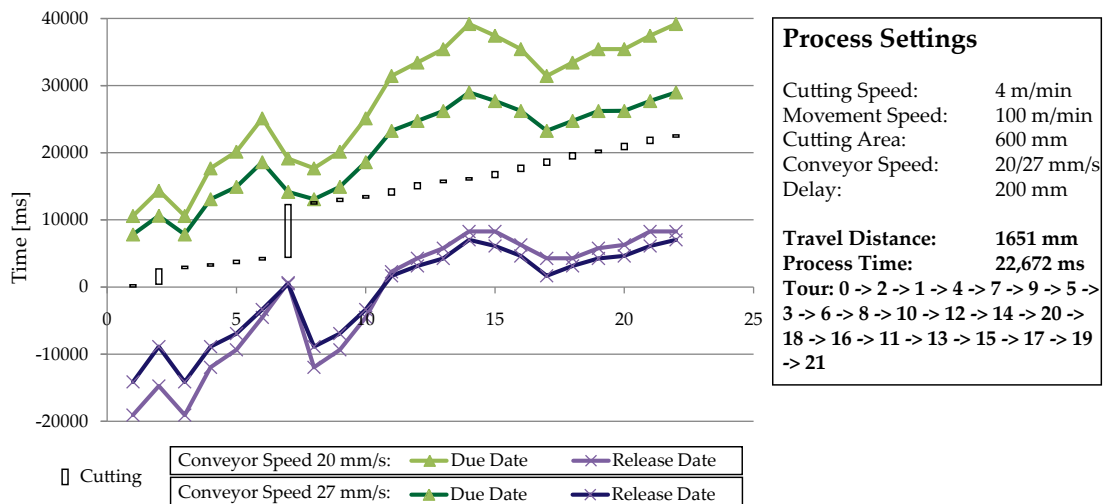


Figure 6.12: Chart showing how the conveyor speed affects the due date and release date line. Both conveyor speeds yields the same path in this example.

Finally, the maximum conveyor speed is estimated by increasing the conveyor speed, until the solver is unable to find a feasible solution to the problem. The conveyor speed is increased to 29 mm/s. It is not possible to increase the conveyor speed further without violating a due date constraint. The cutting area is reduced as much as possible without violating a release date constraint. This gives the result shown in Figure 6.13.

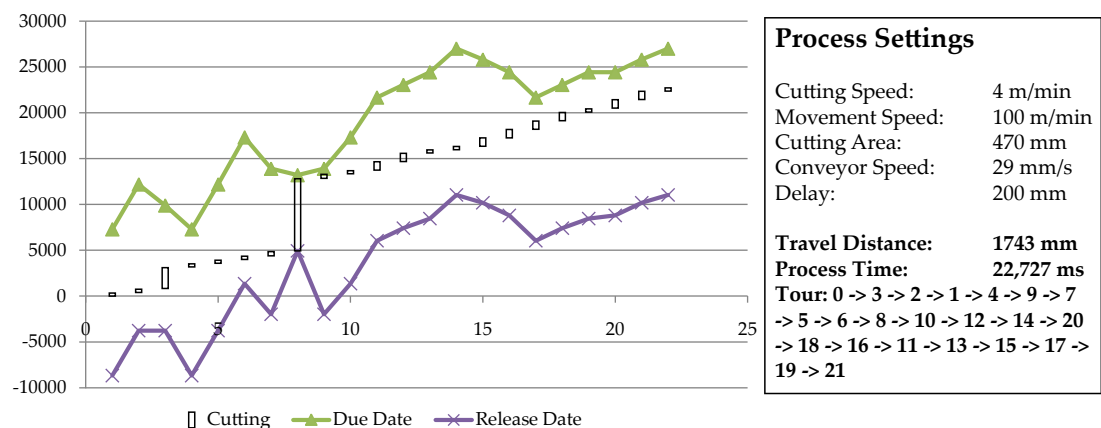


Figure 6.13: Chart showing cutting with CO₂ laser cutter at highest possible conveyor speed and smallest possible cutting area. The delay is 200 mm.

Firstly it is noticed that the path has changed. The travel distance has increased from 1651 mm to 1743 mm. Thus the solver no longer returns the path with the optimum travel distance as this solution would violate the due date and/or the release date constraint. Figure 6.13 shows that task five is limiting the conveyor speed in this scenario. Task five is seen in Figure 6.9(a). Thus, it can be concluded that if task five had a lower cut length the conveyor speed could be increased.

The scenario scheduled in Figure 6.13 assumes a 200 mm delay. If the laser cutter is to cut parts continuously using this schedule it needs to be able to start cutting the next part with a delay of minimum 200 mm as well. If this is not possible the path found cannot be used. In the case shown in Figure 6.13 the difference between the finish time and the due date of the last task is 4293 ms. With a conveyor speed of 29 mm/s this gives a distance of about 124 mm which is not as large as the delay of 200 mm. This indicates that the laser cutter would not be able to commence the cutting of the second part in time. Thus is a conveyor speed of 29 mm/s optimistic.

Scheduling for Remote Laser Cutting

Scheduling with the process data for a remote laser cutter yields the result shown in Figure 6.14.

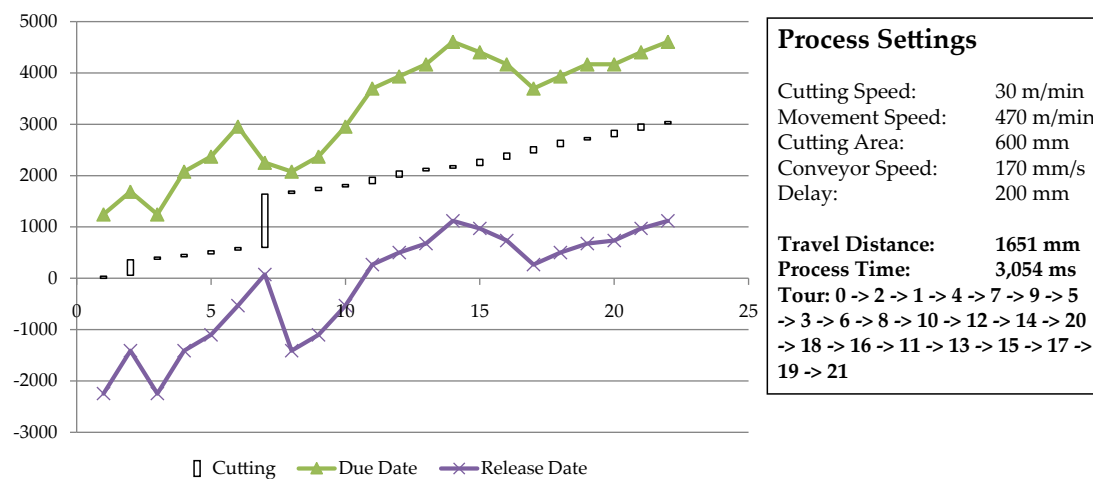


Figure 6.14: Chart showing scheduling results with remote laser cutting process data.

It is obvious that the shapes of the curves are almost identical to Figure 6.11. The paths are identical and the travel distances are therefore also the same. If the conveyor speed is increased and the cutting area is lowered we get the result shown in Figure 6.15. The same ratio is achieved between cutting speed and conveyor speed as we did earlier with the CO₂ laser cutter process data. This means that the cutting area for remote laser cutter will be 470 mm like the CO₂ laser cutter case.

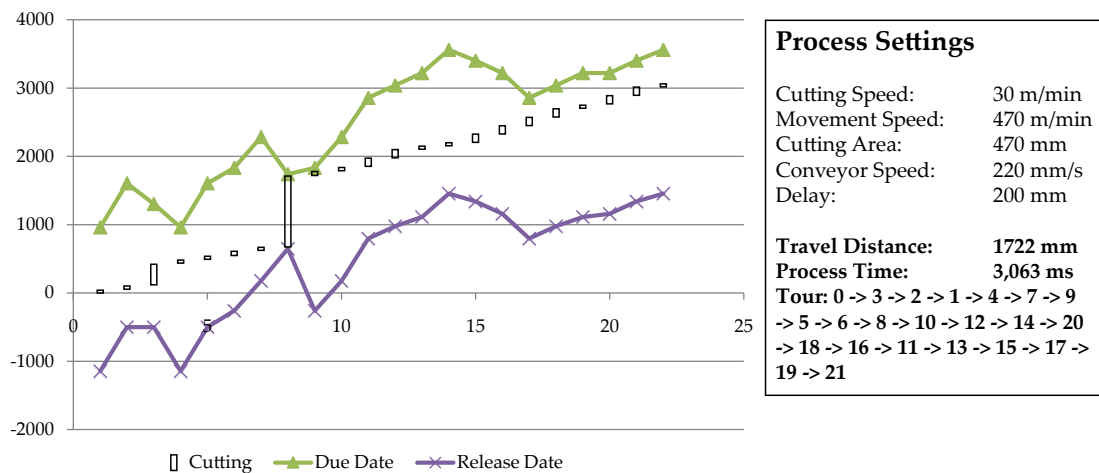


Figure 6.15: Chart showing cutting with remote laser cutting at highest possible conveyor speed and smallest possible cutting area. The delay is 200 mm.

From the results it can be concluded that remote laser cutting with the used process parameters can be carried out with a conveyor speed of more than 170 mm/s.

6.6 Summary

A solver for laser cutting problem is successfully implemented using Gecode. The implementation has given insight to the possibilities of scheduling processes like laser cutting. Especially the limits of the solving techniques have been uncovered. It is clear that if a problem instance contains more than around 25 points the branch-and-bound technique is no longer feasible because of the exponential calculation time as seen in Figure 6.8. In these cases it would be worth looking into construction, improvement heuristics and dispatching rules.

CHAPTER 7

Scheduling Interface

Recalling the proposed Generic RLC Program Generation System from Section 3.1, the scheduling and optimization problem only constitutes a part of the overall problem of converting customer specified CAD-data into machine code and instructions for the production equipment and personnel. Another important part, as it is also seen on Figure 3.2, is the process planning where, among other things, the data and parameters of the scheduling problem is defined. Likewise, there is an also an important step afterwards concerning simulation and visualization of the acquired scheduling results for verification purposes. These two parts together constitute what is defined as the *scheduling interface*.

This chapter will describe a prototype scheduling interface for the human process planner which allows for both the definition of the cutting tasks , process parameters and visualization (but not simulation) of the results acquired from the scheduling software that was developed the two previous chapters.

7.1 Framework

In order to develop a proper interface for the scheduling system, it is necessary to identify and define the overall functional framework of the interface. This is important in order to make an initial decision about what the interface should be composed of.

In this case, the framework is identified by considering the Generic RLC Program Generation System seen in Figure 3.2 as well as considering how the interface can interact with the schedulers that were developed through the two preceding chapters. This yields the framework depicted in Figure 7.1.

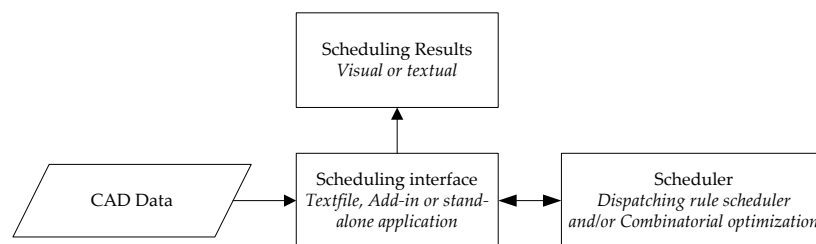


Figure 7.1: Flowchart describing the general framework of the scheduling interface.

The interface acts as the human process planners link between the CAD-data and the scheduler, as well as providing a means of showing the results obtained from the schedulers to the process planner. Based on this basic framework three different interface solutions were identified as described below.

Text files: The simplest interface for the schedulers would be to use formatted text files as the interface only. In this case the process planner would need to manually type in the necessary information pertaining to each cutting task into a text file. This entails that the process planner uses a drawing or some CAD data of the cutting geometries in order to extract this information. The scheduler is then pointed to this text file upon execution and will then afterwards return a the resulting scheduled sequence in a text file. The process planner can then verify the scheduled path by inspecting the text file.

Stand alone application: Instead of having the process planner extracting the information from the CAD-data and then manually typing in this information into text files, this approach instead uses an application that is able to semi-automate parts of the process. The idea is to have an application that is able to load a CAD file and then aid the process planner in the definition of the cutting tasks. Once the tasks have been defined the application passes this on to the scheduler, which may be an external application (i.e. MATLAB script, Gecode executable, etc.) or even completely integrated into the application. Once a result has been obtained, the application could also provide a visual presentation of the results and/or carry out a simulation for verification.

Add-in to CAD-application: Given that the parts and the cutting geometries are commonly designed using some CAD-software, this solution proposes that the definitions of cutting tasks are carried out inside this CAD-environment. This can be done by incorporating the scheduling interface through an Add-In¹. Like the case with the stand-alone application, this could also provide a means of visually showing the results obtained by the scheduler for verification.

Of the above solutions, the interface consisting of text files is the one that presents the least preferable solution, as it requires the most work from the process planner. Furthermore, this solution is also the one that is likely to induce the most errors because the information have to be extracted from the CAD-data and typed in manually. For a small number of cutting tasks this is a manageable task, however this can quickly change as the number of tasks begin to increase. In addition, it is also difficult, and in most cases impossible, for the process planner to verify the scheduling results strictly by inspecting a text file. From these issues it is evident that

¹Add-In: An application that extends the functionality of larger applications.

a software application for aiding and semi-automating parts of the process offers the best solution.

The question is then which of the stand-alone or the add-in application solution can be preferred over the other. This depends on whether the view of consideration is taken in terms of software development or in terms of usability for the process planner.

In terms of implementation the stand-alone application is the preferred choice, as this entails the development of a single application capable of reading CAD file formats that is supported by most CAD applications (e.g. IGES, STEP or DXF). Conversely, the development of an add-in entails the development of an add-in for each of the supported CAD-applications, each having separate API's². The advantage of developing a single add-in is, that it requires less work than the development of a stand-alone application as much of the needed functionality (e.g. environment for visualizing the 3D part) is already present in the CAD application and can be accessed through the API.

Taking the view of usability for the process planner, the add-in is the preferred choice as this incorporates the interface into the CAD-environment that the process planner presumably is accustomed to. At the same time this could also prove to be a more flexible solution, as changes made to the design of the part and the cutting geometries could be taken into account inside the CAD environment automatically. By using a stand-alone application the process planner would have to get accustomed with a new application, but more importantly there could also be problems and restrictions pertaining to the CAD-formats. It should be mentioned that some CAD applications might not offer the possibility of creating a custom add-in, in which case the stand-alone application is the only real solution.

In the case of the example at IAI it was chosen to develop a prototype add-in. The reason being that this offered the best solution in terms of the needed development time. At the same time, the functional ideas and requirements for both the add-in and the stand-alone application would be roughly the same. Thus, an add-in for the CAD-application, *Autodesk Inventor 2011*, used at IAI offers a great demonstration vehicle for testing the idea.

7.2 Requirements

Prior to the development of the prototype, the basic functional requirements for the add-in were identified.

²API: Application Programming Interface is a standard and specification for the interaction between applications.

Define cutting tasks, *by choosing the cutting geometries and specifying the direction of part movement.*

As it was described in Section 4.4 a cutting task is defined by a point and a process time. Thus, it is necessary to define the direction of the part movement to place and orientate the coordinate system in which the points of the tasks are defined. To define the points, and calculate their corresponding process times, each of the cutting shapes (i.e. holes) also needs to be selected. In addition, the user should have the ability to see the holes that have been selected, and only be able to select closed geometries.

Specify process parameters, *by entering the cutting, conveyor and move speeds.*

To calculate the due dates of the tasks as described in Section 4.4.2 as well as the movement time between tasks (setup time), the three process parameters: Cutting, move and conveyor speeds also needs to be specified.

Export cutting tasks and parameters to scheduler, *by exporting to formatted text files.*

Both of the schedulers, developed in Chapters 5 and 6, uses text files as an input. For this reason the information needs to be exported into the two formatted text files: "coordinates.txt" and "constants.txt". Samples of these text files are presented in Section 6.4.3.

Run scheduler, *by referring the user to the scheduler executable/script.*

When the tasks have been exported to the text files the user also needs to be able to execute the scheduler.

Import and visualize results, *by drawing the scheduled path onto the part.*

Once the scheduler has been executed the results needs to be visually presented to the user inside the CAD application for verification purposes. Thus, the add-in needs to parse the file "results.txt", shown previously in Table 6.3. Once the results are parsed, the scheduled path should be shown on the part, while also showing some relevant textual information. In case multiple solutions have been found it should also be possible to "scroll" through them to visually inspect and compare them so the process planner has the ability to choose the best solution.

7.3 The Robocut Add-In

Based on the functional requirements described in the previous section, the prototype "Robocut Add-In" for Autodesk Inventor 2010 was developed. The add-in was developed using the programming language Visual C# and Microsoft Visual Studio 2008 with templates from the Autodesk Inventor SDK Developer Tools [Autodesk Inc., 2011].

The resulting add-in is seen in the screen shot in Figure 7.2, where the example part used in chapters 5 and 6 is loaded showing the defined cutting tasks and a loaded result from the scheduler. The add-in is fully integrated into the Inventor application, and will (once installed) startup together with Inventor, and can be accessed from a button on the toolbar (top right on Figure 7.2).

As it can be seen, the add-in provides an interface for both defining the cutting tasks by defining the movement direction and the cutting shapes (holes) as well as to define the cutting-, conveyor- and move speeds. Furthermore, the add-in can refer the user to the scheduler executable and import the results once they are available. In the shown example a total of eight results have been loaded. Due to some technical issues, it was not possible to implement a function for executing the schedulers from the add-in. Instead the user is lead to the folder containing the scheduler and the exported input files, so the user can run the scheduler manually before returning to using the add-in.

During the development of the add-in, an additional idea for an additional functionality of the interface arose. There are some problems associated with the pre-cutting of holes, described in Appendix A.2, that needs to be taken into account during the design phase and process planning phase. As an example some parts are impossible to roll form if holes are too close to bend lines, or they may cause some undesirable changes of the parts shape and quality. The idea was then to incorporate a functionality for checking that all of the holes can be cut without causing any unwanted defects.

To test this idea, an additional function that checks the selected holes against a simple rule that is entered by the operator was incorporated. As an example, the rule that the distance to the nearest bend line should be at least four times the sheet metal thickness was entered into the system [Halmos, 2006, p. 9-15]. Using this rule, the add-in was detected that the slotted holes (like #10) were too close to the bend lines. As it turns out these holes in fact did cause problems during the roll forming process at IAI [Mathias Theil Petersen, 2011].

The Visual C# project and source code for this add-in can be found in the folder *Part I/Inventor Add-in* on the enclosed CD.

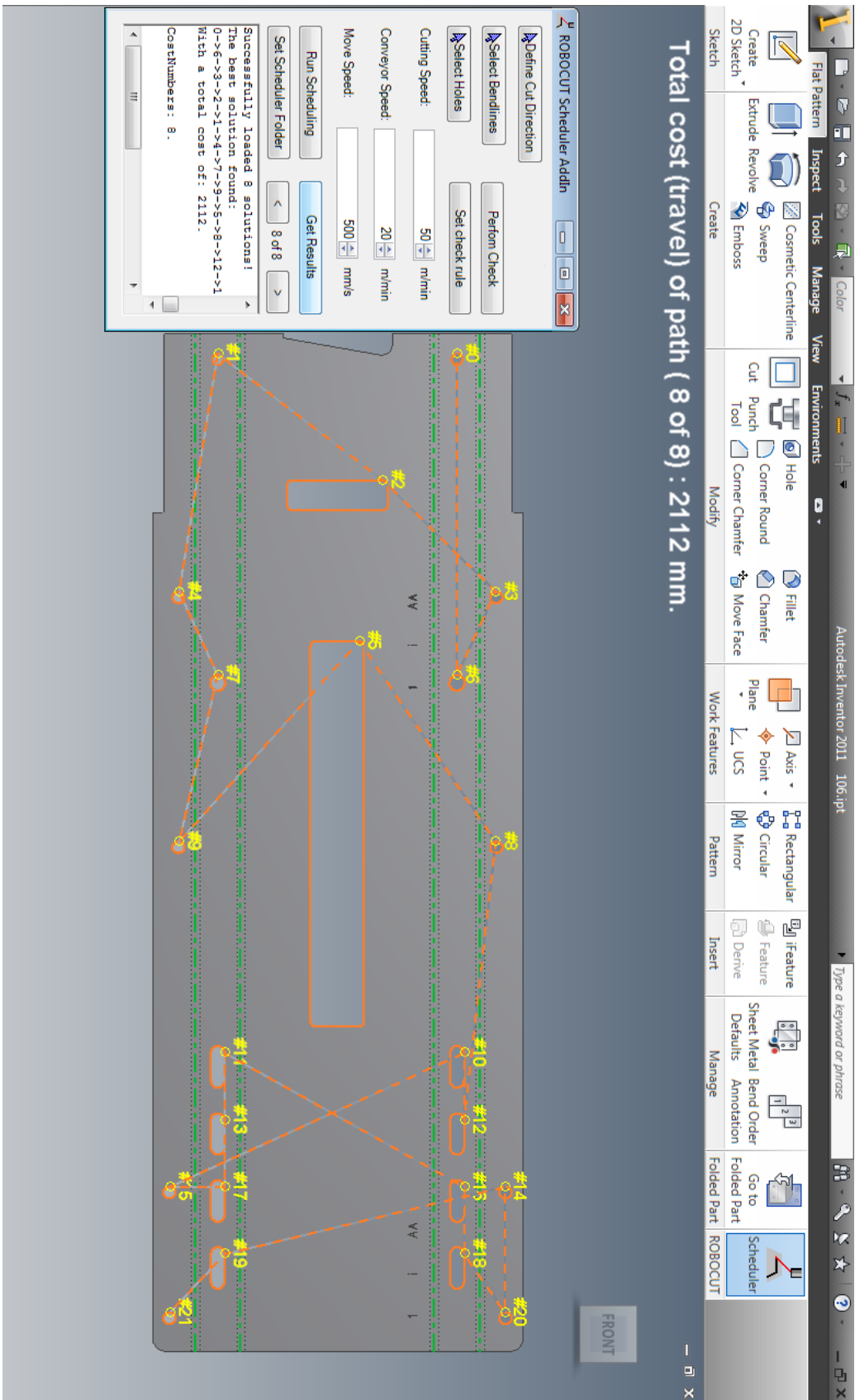


Figure 7.2: Screen shot showing the Robocut Add In inside the Autodesk Inventor Environment. After defining the cutting direction, the bendlines and the holes that should be cut, the scheduler can be executed and the results visualized.

7.4 Summary

In this Chapter an add-in for *Autodesk Inventor 2011* was developed as a scheduling interface prototype. The developed prototype demonstrates how the cutting tasks and parameters can be effectively defined within the CAD-Environment that the human process planner is accustomed to. In addition, it also visualizes the results on the part, thus easing the process of verification.

As a downside, the solution of using an add-in is that entails that one has to be developed for each of the different CAD applications that the users of the RLC technology will have. In addition, the Add-in also offers little possibility of extending the functionality to include simulation as well, as this requires some elements that can not be guaranteed to be present in the CAD applications (e.g. robot dynamics).

CHAPTER 8

Discussion

The laser cutting problem has been solved using mainly two approaches in this chapter. The first approach presented in Chapter 5 applied the general theory of dispatching rules. Using the basic dispatching rules showed to return some paths that were not able to satisfy multiple objectives. To overcome this problem the combination of basic dispatching rules into composite dispatching rules were studied, specifically the ACTS rule was studied. This demonstrated the difficulties of both the construction and especially the choice of parameters for the scaling of individual terms of composite dispatching rules.

The ATCS rule was applied using two approaches were studied for determining the scaling parameters; statistics and trials. The latter approach showed some good results, however with an additional cost in computation time. Even so, this calculation time was still significantly lower than that of combinatorial optimization. This however is based on a fixed search space of the parameters for a specific task set. As the search may change with the task set, the necessary computation time will also change. In addition to this the search space this was determined through trial runs. Together this means that there is a possibility of improving the trials approach by introducing some optimization techniques for first determining the search space and then carry out a targeted search (e.g. Steepest Descent, Least Square, etc.) for the optimal parameters. This could yield some improvement both in terms of the obtained solutions (by fine tuning the search space) but also in terms of reduced computation time (through a targeted search).

In terms of the results obtained through the dispatching rule scheduler, they were obtained based on the due dates, processing times and setup times only. In effect, the dispatching rules did not take the release date, which is calculated based on the possible cutting area, into account. For this reason the solutions obtained by the dispatching rule scheduler are only valid for large cutting areas. Thus to obtain some results that are realistic in terms of the actual cutting area of the scanner head, the release dates should also be taken into account in the basic dispatching rules like it was done for combinatorial optimization.

The combinatorial optimization presented in Chapter 6 applied the mathematical theories of combinatorial optimization in order to find the optimal solution to the

laser cutting problem. This approach showed to be effective for the example part used, which contains 22 tasks. A study of the calculation time however showed that this approach is not scalable. If this approach is used for parts containing more than around 25 tasks, there are two options. The first possibility is to divide the tasks into groups. In this way the optimum path between the groups can be calculated followed by finding the optimum path between the tasks in each group. The second possibility is to use a heuristics that can handle the high number of tasks.

Which of these two approaches is then preferable? Both approaches have potential to yield quality paths. If heuristics are used it can be concluded that path construction heuristics cannot be trusted to yield quality paths alone. Note that construction heuristics also covers dispatching rules. This problem can be overcome by implementing an improvement heuristic. Improvement heuristics can provide quality paths for large problems. But implementing an effective improvement heuristics is difficult and there are countless ways to tackle the problem.

If the tasks are divided into groups the branch-and-bound method implemented can be used for finding an optimal path within each group. Solution methods to the Generalized Traveling Salesman Problem (GTSP) can find the optimum path between the groups. Finding optimal paths between groups of tasks has been successfully used in Stemmann and Zunke [2006] for robot scheduling with shorter cycle times as a result. If using this approach for solving the laser cutting an algorithm that divides the tasks into groups is needed.

The most straightforward approach for solving large instances based on the work done in this thesis would be to divide the tasks into groups and focus on implementing the GTSP for obtaining the optimal path between the groups. Finding optimum solutions to the GTSP using branch-and-bound is possible. It is however uncertain how large problem sizes a branch-and-bound solver would be able to handle when solving the GTSP. Using exact algorithms the GTSP has been solved for instances with up to 89 groups and 442 nodes in the literature [Pop et al., 2010, p. 63]. If it turns out that the branch-and-bound solver cannot solve the GTSP within an acceptable time it could be solved using heuristics.

A crucial function still needs to be implemented in the final branch-and-bound solver. If at some point during the process the laser cutter does not have any geometries to cut in the cutting area the solver will not return any solutions. The solver should allow the laser cutter to "pause" until a new task has entered the cutting area and then start cutting.

Currently the branch-and-bound solver minimizes the cost of the path while ensuring due dates and release dates are met. A number of other constraints could be implemented like the cutting angle. Adding additional constraint could however cause problems. If one of the imposed constraints cannot be met the solver will not

output any solutions. The solver cannot inform about what constraints are not met and the user is therefore struggling blindly in order to find out what constraints or process parameters are causing problems. The more constraints are added the more difficult it will be to see why the solver cannot find any feasible solutions.

PART
II
ROBOCUT CONFIGURATOR

CHAPTER 9

Introduction

The ROBOCUT technology has potential to be used in countless production scenarios. This potential can only be utilized if producers are aware of the existence of the technology and understand its capabilities. This part of the thesis will focus on developing a production system configurator that should convey information about the ROBOCUT technology to producers.

Online product configurators systems are used extensively on the internet to allow customers to add and change functionalities of a core product. The core product could be a BMW car, a bicycle, a computer etc. These online product configurators can be considered an implementation of mass customization.

The concept of mass customization was coined in the early 2000s and is the method of "providing high volume products that are individually customized to meet the specific needs of each customer" [Davis and Heineke, 2005, p. 31]. A very important part of the evolution of mass customization is the advances in technology, especially the internet. Producers are able to provide customers with easy to use online configurators that can be directly coupled with the production system. The technology has had a great influence on the development of mass customization.

Mass customization among other tendencies has increased the need of flexibility and responsive productions. The trend today in order to meet this need is Reconfigurable Manufacturing Systems (RMS). The concept of RMS is to modularize manufacturing system components, controllers, machine tools, etc. This modularization should then facilitate the possibility of tailoring one specific module configuration for each company's production needs [Jørgensen et al., 2010, p. 1]. This modularization should also make it possible to quickly adjust production capacity and functionality in order to meet sudden changes in the market [Bi et al., 2008, p. 974].

In order to construct a configurator for the ROBOCUT technology a production unit based on the ROBOCUT technology is defined along with a clear definition of modules and their interfaces. This is a way of considering the ROBOCUT technology in context of the RMS paradigm. This is done in Chapter 10.

Developing the configurator software is a process that involves a set of activities and results that in the end produce the software product. The scope of the thesis does

not allow for a documentation of all steps. The four fundamental process activities that are common to all software processes are showed in Figure 9.1.

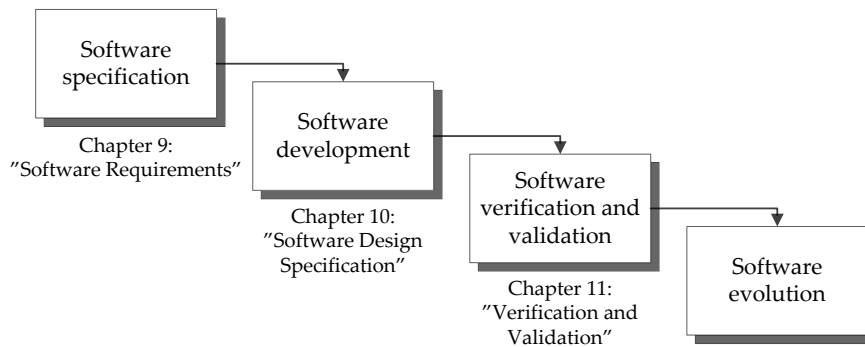


Figure 9.1: *The four fundamental process activities associated with software development [Sommerville, 2006, p. 8].*

Figure 9.1 shows that the activities in "Software specification" is covered in Chapter 11, "Software development" is covered in Chapter 12 and "Software verification and validation" is covered in Chapter 13. Each chapter gives an introduction to the software process it covers.

"Software evolution" is not covered in this thesis as it refers to the process of modifying software to adapt it to changing customer and market requirements. In order words software evolution deals with the development that takes places *after* the software is deployed and taken into use.

9.1 Notation

This part of the report will frequently refer to modules and variants. Figure 9.2 shows the notation used in this thesis. A production system consists of one or more production units that consist of one or more modules that is of a specific variant. The vertical lines to the right of the diagram indicate the cardinality.

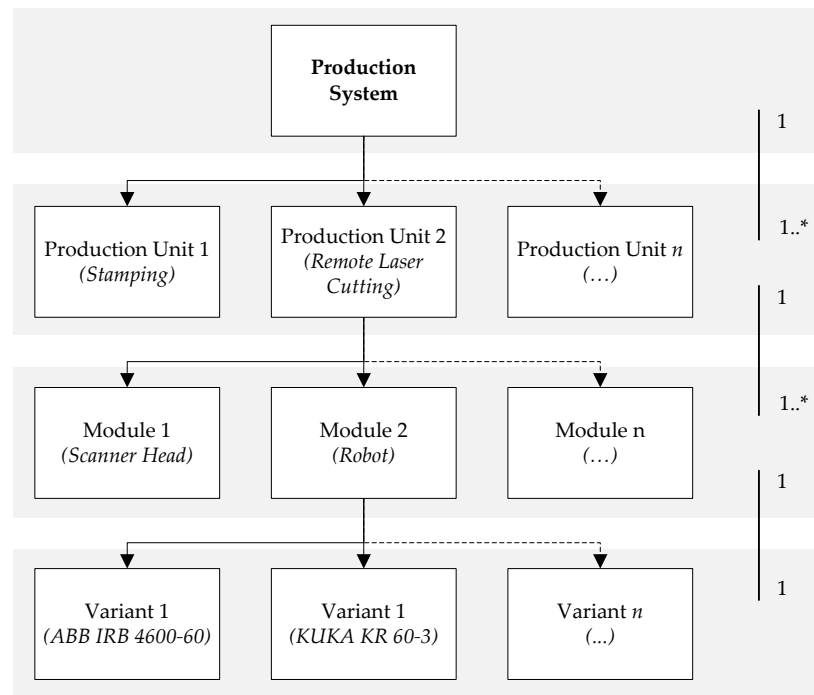


Figure 9.2: Relationship between modules and variants with indication of cardinality.

Identifying Modular Architecture

The identification and design of the modular architecture for configuration of the RLC/RLW production unit is based on the methodology presented in Pahl et al. [2007] for the design of modular products. The five step methodology, seen in Figure 10.1, have been modified by changing the scope of the first four steps to fit the modularization of a production unit, and by changing the fifth step from being a preparation of the modular product layout to instead being a preparation of the modular architecture for deployment.

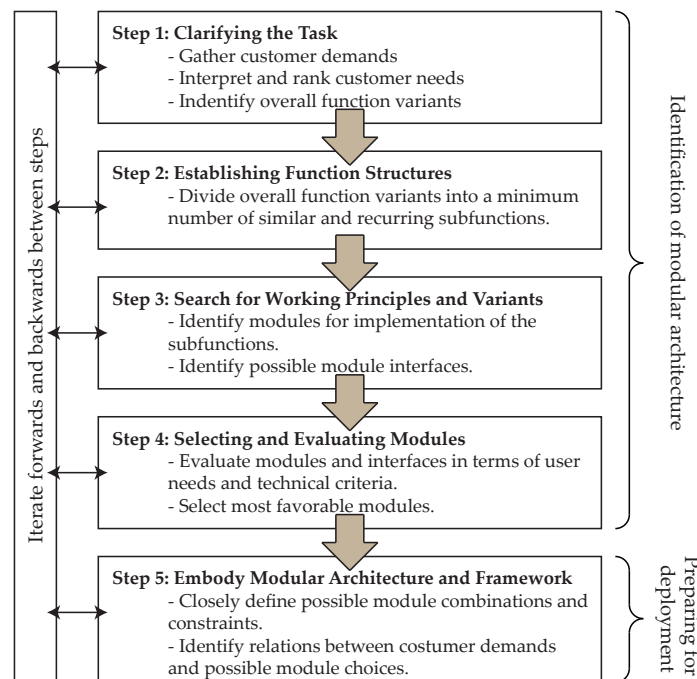


Figure 10.1: The methodology used for identifying and preparing the modular architecture of the production unit for configuration.

The following will explain each individual step of the methodology based on Figure 10.1.

In *Step 1: Clarifying the Task*, the overall functionality of the production unit is defined. By gathering information from the potential customers (manufacturing companies) of the production unit, some possible uses and functional demands for

the RLC/RLW production unit are also identified. This information is then interpreted into specific functional needs and ranked in terms of importance.

The overall function is then divided into a number of generic subfunctions in *Step 2: Establishing Function Structures*. This should be done based on the customer demands, while trying to identify a minimum amount of subfunctions that are similar and/or recurring. Also, the subfunctions should be identified such that they can be combined as easily as possible to implement the required overall functionality.

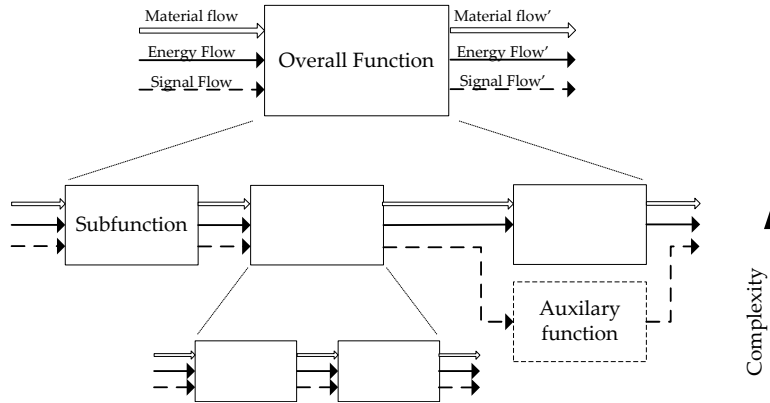


Figure 10.2: Functional Modeling Method. Functions are described as the conversion of materials, energy and signals. The overall function is divided into subfunctions to create a function structure. Notice that a subdivision of a function decreases the level of complexity [Pahl et al., 2007, p. 32].

To better understand the breaking down in functions carried out in the first two steps, the graphical functional modeling method presented in Pahl et al. [2007] is used. The method, illustrated on Figure 10.2, defines a function as the conversion of materials, energy and signals and describes it based on its inputs and outputs [Pahl et al., 2007, p. 30]. A similar approach is also proposed by Ulrich and Eppinger [2004], however this is less detailed.

A search for possible module types for implementation of the identified subfunctions is then carried out in *Step 3: Search for Working Principles and Variants*. While performing this search, the possible interactions and relations between the individual modules (subfunctions) should also be explored. It is important to note that as the functional needs specified by the customers will vary, it might not be possible to select a single module type for a given subfunction capable of satisfying the needs in all cases. In such cases, multiple possibilities for the same subfunction should be chosen, ensure that the entire possible scope of customer needs and requirements is covered.

In *Step 4: Selecting and Evaluating Modules*, a search and selection of specific module variants is carried out based on the identified module types, as well as the functional needs. The module variants are selected by evaluation against the functional needs and requirements found in step one.

Finally, in *Step 5: Embody Modular Architecture and framework* the selected module variants are put together to form a modular architecture. In this last step the possible combinations of the module types and variants are defined. As some of the module types and variants might be incompatible, or some combinations simply makes no sense, the constraints and possible combinations are also defined. Once the possible combinations have been clarified, the modules are related to the customer demands with the objective of determining how the individual modules can be chosen to form a complete configuration.

During the execution of these five stages, new knowledge and information can make it likely to make it necessary to iterate forwards and backwards between the stages.

10.1 Clarifying the Task

Prior to beginning the identification of the modular architecture, it is necessary to gain a more detailed overview of the task. This entails a description of the overall expected functionality of the production unit (in this case the RLC/RLW production unit), as well as gathering and interpreting customer demands to specify goal functionalities and specifications.

10.1.1 The Basic Functionality

The ROBOCUT project aims at developing a new remote laser cutting and welding process. Thus, this is the overall functionality of the production unit, as depicted in Figure 10.3.

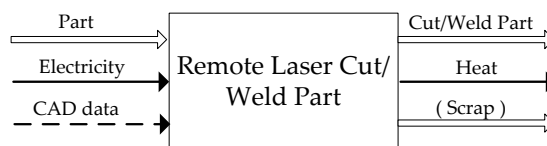


Figure 10.3: The overall functionality of the remote laser cutting/welding production unit.

As input, the function gets a part (material) along with the CAD data (signal) about the part and the cut/weld path and finally some electricity (energy) to perform the cut/weld. The function (i.e. RLC/RLW) then converts these inputs, yielding a cut/welded part (material) and heat (energy). In case the function has performed a cut, the function will also return some scrap (material).

The RLC/RLW production unit is going to be used in different application scenarios. This means that the subfunctions and function structures will change,

depending on the customers (manufacturing companies), while still having the same overall functionality.

As described in Section 4.1, the RLC/RLW production unit consists of the scanner head for positioning and focusing the laser beam source coming from a high power fiber laser. Because the scanner head will have some limited working area, it may need to be moved around the part to extend the working area as well as to reach different surfaces of the part. On the other hand it is also possible that the part needs to be moved instead. It might even be the case that both the scanner head and the part needs to be moved relative to each other.

10.1.2 Customer Specific Functional Requirements

Different customer needs and application scenarios will lead to different subfunctions and function structures (e.g. some customers need a subfunction for moving the part while others do not). For this reason it is necessary to examine the needs and requirements of some potential customers for the RLC/RLW production unit. This also provides a better understanding of the different possible usage scenarios of the production unit. This survey also serves the purpose of identifying the different and overlapping needs and requirements that the production unit should be able to perform.

In this case, the primary customers are the manufacturing companies working with the machining and shaping of metals. These companies are a part of the metal products industry. In Denmark these counted around 3200 with a total revenue of more than 54 billion DKK in 2008 [Danmarks Statistik, 2011]. This, together with the business potential described in Section 1.1.1 makes for a large, but also diverse, customer segment.

Gathering and identifying the customer needs is a process in itself, and it is important to ensure that it is carried out as good and effectively as possible in order to capture all explicit as well as hidden needs of the customers. A structured five-step approach to ensure this is proposed by Ulrich and Eppinger [2004]. In this case however, three customers are already involved in the development of the RLC/RLW process. Thus, their needs and requirements for the production unit have already been described through the ROBOCUT project. For this reason those needs are used, instead of carrying out the structured five-step approach.

In what follows, a total of five application scenarios in the three companies are briefly described. The application scenarios presented only represent the use of the RLC production unit. This is because the needs collected through the ROBOCUT project focuses solely on the RLC process, even though RLW is a part of the project. However, this is assessed not to be a problem, as the functional needs and

requirements of the RLC and RLW production units is expected to be similar. It is also important to note that the needs have been collected with a different goal in mind, than that of identifying a modular architecture of the RLC/RLW production unit. As a result, some information about the intended applications of the RLC/RLW production unit are lacking. For this reason it has been necessary to fill in the gaps by making an assessment of the expected application scenarios. On a final note the requirements have not been ranked in terms of their importance, as the methodology prescribes, because of an insufficient level of detail in the available information. Instead the requirements are all assumed to be equally important, with some exceptions that are marked as soft specifications.

10.1.3 Ib Andresen Industri

As described in Appendix A.2, Ib Andresen Industri A/S (IAI) wants to use the RLC process for the cutting of holes on the fly in the roll forming lines. IAI has two possible usage scenarios of the RLC process in mind; (1) Ahead of or inside the roll forming mill and (2) after the roll forming mill.

Cutting of holes on the fly ahead of or inside the roll forming mill

This scenario is the one that was also treated in Part I. IAI wants to place the RLC scanner head in between sets of rolls, as sketched in Figure 10.4.

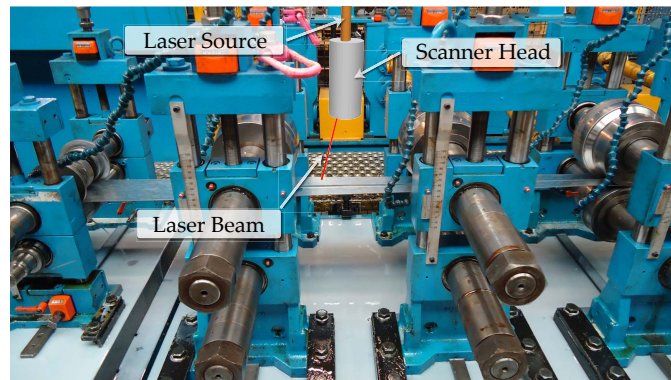


Figure 10.4: Close-up of a set of rolls in a roll forming mill at Ib Andresen Industri. The scanner head could potentially be placed in between a set of rolls to perform the cutting on the fly.

As it was explained in Section 4.2, a metal strip passes underneath the scanner head continuously while the scanner head cuts the holes. The scanner head can be statically mounted or moved sideways across the strip. This depends on the size of the scanner heads working area as well as the width of the metal strip, or more correctly the maximum width between cut geometries. Besides cutting top-down, as the case on

Figure 10.4, IAI also wants the possibility of mounting the scanner head pointing in other directions (e.g. from the sides or upside down) [IPU, 2011, pp. 7-8].

Cutting of holes after the roll forming mill

Due to some problems related to the cutting of some holes ahead of roll forming, described in detail in Appendix A.2, IAI could also find use of the RLC process after the roll forming mill. In this application, the part has been roll formed, like the example given in Figure A.5 in Appendix, and is about to be cut to length. Thus, the holes to cut are not necessarily located in a single plane as the case described above. In turn this means that the scanner head needs to move around the part in order to cut holes in different surfaces of the part (or even around corners) [IPU, 2011, pp. 7-8].

IAI's needs and requirements:

Based on the described application scenarios, IAI's technical and functional requirements for the RLC process can be summarized as shown in Table 10.1.

Process-specific requirements:
Material: Stainless steel up 3 mm in thickness.
Cut tolerances: ± 0.1 mm.
Cutting rates: ≈ 15 m/min.
* Scanner head cutting field: 250x100 mm.
Positioning of scanner head: with a precision of ± 0.2 mm.
Functional requirements:
Placement of scanner head between rolls.
Adjustable for cutting from all angles.
Cutting in one plane from a static position.
Cutting around corners with movement of scanner head.
* On-the-fly cutting.
* <i>Soft requirement</i>

Table 10.1: IAI requirements for the RLC process [IPU, 2011, pp. 7-8].

10.1.4 Grundfos

The pump manufacturing company Grundfos A/S, located in Bjerringbro, Denmark, is also interested in using the RLC process in their factories. For this reason they are also involved in the ROBOCUT project. The exact application scenario(s), parts, and cutting geometries they have in mind are unknown, but two possible scenarios have been identified.

Robotic RLC of metal parts

Grundfos plans to mount the scanner head on a robot, like the situation seen in Figure 1.1(a) [IPU, 2011, pp. 5-6]. In this way, the scanner head performs the detailed positioning of the laser beam while the robot moves it in a smooth path around the part to cut. This will give the advantage of an increased speed, when compared to the conventional robotic laser cutting (see Figure 10.5), where the robot needs to perform the detailed positioning of the laser beam as well.

It is assumed that the part also needs to be moved in a plane, by an XY-table during the cutting process. There is a possibility, that the parts will remain stationary during the cutting process, in which case the parts will only need to be loaded somehow - either manually or automatically, so that situation should also be kept in mind. To ease the programming of the robot, Grundfos wish to have the control of the scannerhead (i.e. focus and position of the laser beam) fully integrated into the robot controller. In addition Grundfos would like the weight of the scanner head not to exceed 40 kg, to limit the necessary size of the robot.

Cutting of holes at a station

Because of the RLC's capability to cut holes within some working area from a static position, it drastically reduces the spacial requirements. This makes it possible to integrate a small station for laser cutting smaller parts into a continuous production line. As Grundfos has many such lines producing smaller parts for pumps, it is assumed that they could also use the RLC process as a station in a production line. To further explain, it is assumed that the scanner head is statically mounted and that the part is placed beneath it, then cut, and then passed on for further processing down the production line. In this case both the scanner head and the part remains stationary during the process.

Process-specific requirements:

- Material: Stainless Steel 0.5 - 2.5 mm in thickness.
- Repeatability (absolute tolerance): ± 0.05 mm.
- * Cutting Rates: 100 m/min in 1 mm stainless steel.
- Scanner head cutting field: 200x200 mm.
- Cutting angle: Minimum 45° but preferably 60°.

Functional requirements:

- ** Movement of part in a plane.
- Movement of scanner head with robot.
- Robot-controller integration of scanner head.
- Scanner head weight maximum 40 kg.
- ** Cutting from static position.

* Soft requirement ** Assumed requirement.

Table 10.2: Grundfos requirements for the RLC process [IPU, 2011, pp. 5-6].

Grundfos requirements

Based on the above described application scenarios, as well as the requirements stated by Grundfos [IPU, 2011, pp. 5..6], the technical and functional requirements for the RLC process can be summarized as shown in Table 10.2.

10.1.5 Volvo

The car manufacturer, Volvo Car Corporation (VCC), is also a part of the ROBOCUT project, as they wish to utilize the RLC process for hole cutting on pre-painted car bodies, and possibly other parts. Currently the laser cutting of holes is performed with the conventional robotic laser cutting process, as it is seen on Figure 10.5. VCC plans to move the laser cutting of certain variant specific holes (e.g. holes for accessories) from the body shop to the assembly line, and this should be made possible by the increased flexibility offered by the RLC process [Volvo Car Corporation, 2011].

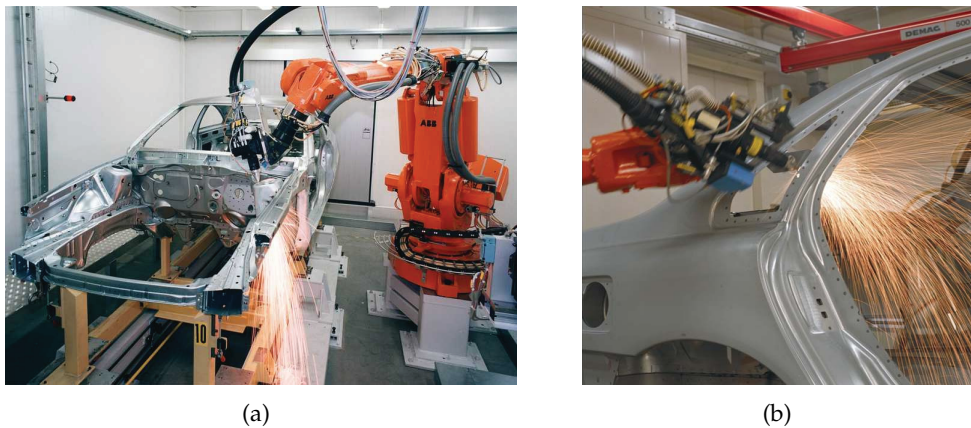


Figure 10.5: *The current robotic laser cutting at Volvo [Volvo Car Corporation, 2011].*

As indicated on Figure 10.5(a) the scanner head will presumably be mounted on a robot, as a car body can be considered a rather complex part with some hard to reach areas. It is also assumed that the part (i.e. car body) will remain stationary during the cutting process, as also indicated by Figure 10.5(a).

In this application scenario the cutting quality and the accuracy of the process are important, however there is one additional functional requirement that needs to be taken into account and that is the removal of scrap. When cutting holes in a car body, it is important to ensure that the scrap from the cutting process is not trapped inside hollow sections. In the current process seen up close in Figure 10.5(b) this is handled by a suction nozzle placed beside the cutting nozzle of the laser cutting head. In this way, the scrap metal is sucked out of the cut hole. This approach however is not

possible with the RLC process as the cut is performed remotely. In turn, some other means of scrap removal needs to be engineered for the application of RLC at VCC.

Volvo requirements

As VCC has not specified any technical requirements, the requirements for the application scenario described above can be summarized as shown in Table 10.3:

Functional requirements:

Part always stationary during cutting.
Movement of scanner head with robot.
Scrap-removal functionality.

Table 10.3: VCC's requirements for the RLC process.

10.1.6 Technical Goal Specifications:

Besides the functional and Process-specific requirements specified by the companies (customers), there also exists some technical goal specifications for the RLC and RLW technology that has been specified in the ROBOCUT project. These specifications are listed in Table 10.4.

Technical specifications:

Laser Power: 2 - 4 kW.
Beam Wavelength: 1.06 - 1.09 μm
Beam Quality: 1.0 - 1.2 M^2
Fiber Diameter: 20 - 50 μm
Effective focal length: 200 - 300 mm.
Cooling: Water cooling.
Working area: 100 x 100 x 100 mm.
Maximum beam travel speed: 90 m/min.

Table 10.4: Preliminary ROBOCUT specifications [Olsen, 2011, p. 6].

10.2 Establishing Function Structures

For each of the five application scenarios described in the previous section, a decomposition of the overall function into a number of subfunctions and function structure is performed. The goal for performing this decomposition is to identify a generic function structure that is applicable to all of the possible application scenarios, such that the minimum amount of subfunctions and thus modules can be identified [Pahl et al., 2007, p. 500].

Conventionally the division of the overall function into subfunctions is performed in a very detailed manner, in order to identify subfunctions that should form a single module [Pahl et al., 2007, p. 503]. In this case however the production unit will consist of modules that are already defined or available, but haven't been identified yet. Put in other words, it is not possible to control the composition of the individual modules. Instead the general subfunctions should be identified in a way such that it is possible to find existing modules (in Section 10.3) to carry them out.

As previously described, the functional modeling approach shown in Figure 10.2 is used to visualize the function structure. Two of the five scenarios will be described here in detail, to demonstrate the division into subfunctions and function structure. The reader is encouraged to review Appendix D, where all of the function structures can be found in a larger format.

10.2.1 Function Structure for RLC ahead of or inside the roll forming mill at IAI

Based on the description given in section 10.1.3 the function structure diagram seen in Figure 10.6 was constructed. Recall that the case under consideration is also the one that was described in Section 4.2.

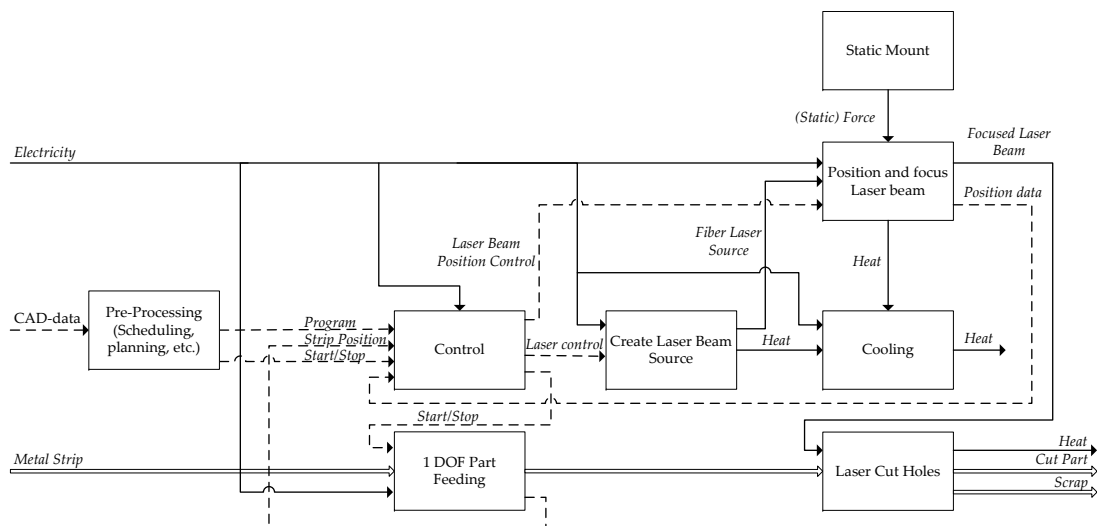


Figure 10.6: Functional structure for RLC of holes ahead of or inside the roll forming mill at IAI.

Going from left to right, a program for the process controller is constructed through some software, like the RLC/RLW program generation system described in Section 3.1. Once created, the program (signal) is passed on to the process controller for continuous execution for as long as the current part needs to be produced. Through the controller's execution of the program, the position and focus of the laser beam,

the laser source and the movement of the metal strip is controlled, based on their respective feedback signals.

The function *Create Laser Beam Source*, representing a fiber laser, creates and returns a laser source (energy) according to the signal received from the controller. This function will also return some excess heat, which is handled by the *Cooling* function.

To use the laser source, it needs to be positioned and focused first, handled by the function *Position and Focus Laser beam*, representing the scanner head. This function also creates some excess heat that is passed on to the *Cooling* function. In the current case the scanner head needs to be mounted in a static position, done by the *Static Mount* function. Notice that the force transferred from this function only represents a static force (from gravity), and that it is included to follow the functional modeling method.

Simultaneously, the function *1 DOF Part Feeding*, representing the roll forming mill, moves the metal strip, while returning the position to the controller. The moving metal is then passed on to the function *Laser Cut Holes*, where the laser cutting process takes place. The result is a metal strip with holes that are cut as specified by the CAD-data, along with some scrap metal and excess heat.

10.2.2 Function Structure for Robotic RLC at Grundfos

The function structure diagram for the first Grundfos application scenario, described in section 10.1.4, is seen in Figure 10.7.

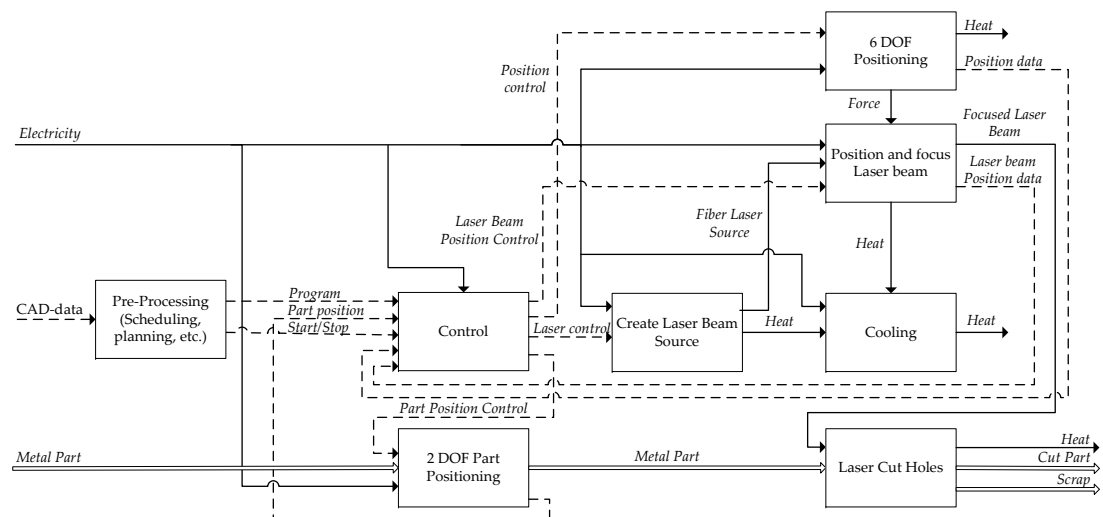


Figure 10.7: Functional structure for robotic RLC at Grundfos.

By comparing Figure 10.7 with Figure 10.6 it is apparent that this and the previous application share some common functions. In this case however the scanner head

needs to be moved by a robot, represented by the function *6 DOF Positioning*. The movement is performed based on a control signal from the controller which in turn receives a feedback signal with position data.

In this case, the part also needs to be moved in a plane. This is performed by the function *2 DOF Part Positioning*. Again, this is performed based on a control signal coming from the controller, and a feedback signal with the current position of the part is also returned to the controller.

10.2.3 Generic Function Structure

Observing the function structures in Figures 10.6 and 10.7 it can be seen that the two separate applications share some common functions. This is also expected, as the fiber laser and the scanner head are always necessary independently of the application scenario. The surrounding system however will change for different application scenarios.

By comparing all the function structures, found in Appendix D, the generic function structure seen in Figure 10.8 was composed.

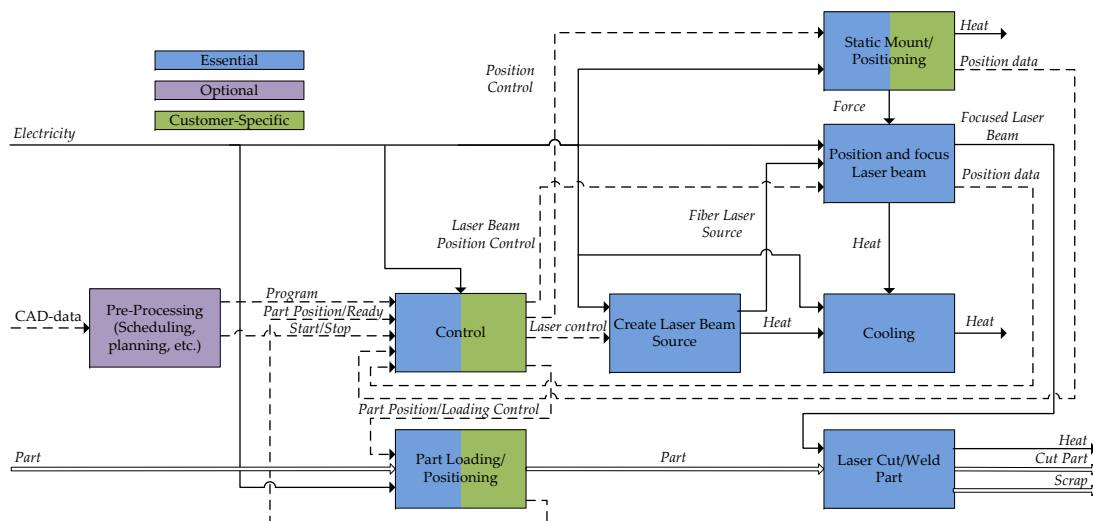


Figure 10.8: Generic functional structure of the RLC process.

Each of the subfunctions has been classified as being either *Essential*, *Optional* and *Customer-Specific* functions [Pahl et al., 2007, p. 496]. Notice that some functions have two classifications. The essential functions, as the name implies, is always a part of the RLC production unit contrary to the optional functions. Customer-specific means that it varies significantly between customers and applications. That is, there is a significant difference in the working principle. There is also a possibility that these functions are completely supplied by the customer (like the roll forming mill at IAI that moves the part).

As seen on Figure 10.8, only the three functions *Static Mount/Positioning*, *Controller* and *Part Loading/Positioning* varies significantly between customers and applications, and they are all essential modules as well. Hence these functions is the most important in customizing the RLC production unit to the individual customer. For the same reason these functions are chosen to be individual modules, even though they might affect each other. In addition to this, the optional *Pre-processing* function will be included as a part of the Controller module, as they are related. The reason being that the pre-processing software and its requirements may change with the choice of controller since they have different capabilities and attributes (e.g. programming language and software).

The three functions *Position and focus Laser beam*, *Create Laser Beam Source* and *Cooling* are all essential functions that does not vary significantly in their working principle, as will be explained in Section 10.3. For this reason, these functions will be modified for the individual customer and application together, as a single module.

Notice that the *Laser Cut/Weld Part* function is not considered a module, as this merely represents the process taking place.

To summarize the modules identified from the functional structure of the RLC production unit are repeated and named below:

- ◇ *Static Mount/Positioning*: Movement of scanner head.
- ◇ *Part Loading/Positioning*: Movement of part.
- ◇ *Control and Pre-processing*: Controller.
- ◇ *Create Laser Beam Source, Cooling and Position and focus Laser beam*: Fiber Laser and Scanner Head (with DOE).

10.3 Working Principles and Variants

With the generic function structure identified, the search for different possible working principles for each of the identified subfunctions can be carried out. In this case the working principles are defined as existing components that can carry out the subfunctions, either alone or in combination with other components. The search will be carried out for each module separately, such that working principle(s) for all subfunctions of a module are identified together.

10.3.1 Movement of Scanner Head

By considering the function structures in Figures 10.6 and 10.7, as well as Figures D.1 through D.5 it is observed that the functional requirements spans from no movement to movement with six *degrees of freedom* (DOF).

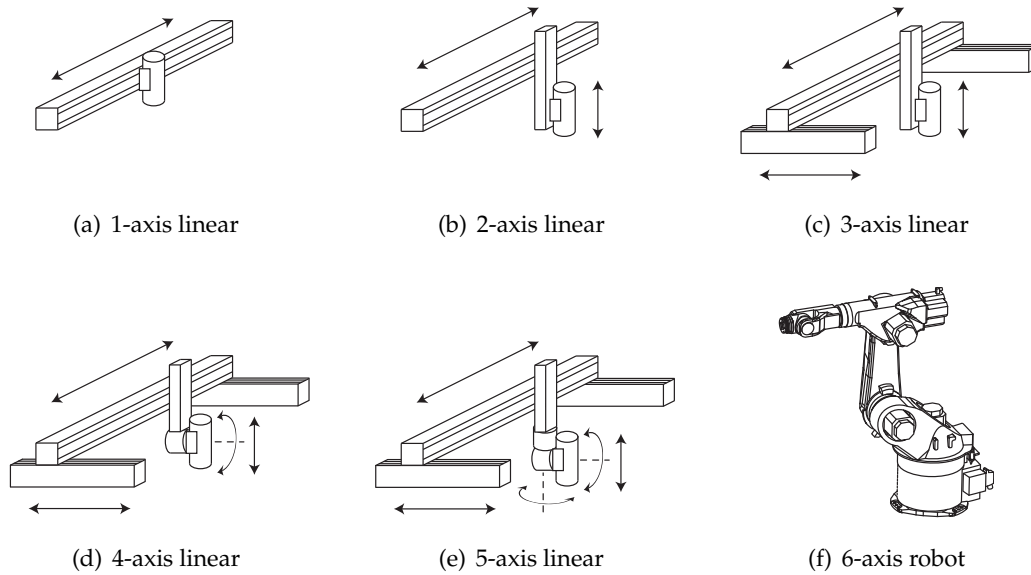


Figure 10.9: The found working principles for moving the scanner head.

Based on this observation, as well as the functional requirements and specifications given in Section 10.1.2, the working principles shown in Figure 10.9 were identified for the movement of the scanner head:

Static Mount, *i.e. no movement*. In this case the scanner head will be mounted either using a stand-alone bracket designed for this purpose, or through a bracket integrated into the scanner head itself.

Linear modules, *for movement in up to five DOF*. In case the shape to cut/weld lies on a plane, but exceeds the working area of the scanner head, the scanner head can be moved in either one (a) or two (b) DOF. For parts with geometries in planes with different orientations, up to three (c) or four (d) DOF are typically needed. Finally, for the complex geometries, with convex/concave surfaces, as much as five (e) DOF are needed. It is important to note that none of these modules represent absolute choices. Hence, five DOF may be chosen even though only two DOF are needed.

Robot, *for complex movement in six DOF*. In some cases a linear movement of the scanner head is not possible or insufficient, thus needing a movement in six

DOF with a robot (f). For instance, this is the case for the car body seen in Figure 10.5(a), where some cuts/welds might lie inside the car body, making them practically unreachable for linear modules. Generally, the robot is more agile in its ability to reach the same position in different ways. Though, it should be noted, that the sixth DOF is actually redundant, since the scanner head has two internal DOF with a rotary symmetric working area. In case the robot has an insufficient reach it can be combined with a linear gantry, allowing the robot to traverse alongside the part.

It is important to note that the working principles have been selected by screening the possibilities available from suppliers, and only choosing those that is able to satisfy the functional and technical requirements and specifications that were found in Section 10.1.2. As an example, the movement of the scanner head with a fewer DOF robot were also investigated as supplementary choices to the linear modules in Figures 10.9(c) and 10.9(e). It turned out that that these types of robots are generally not able to carry the expected weight of the scanner head (up to 40 kg) or are intended for other applications than cutting and/or welding.

All of the working principles described above are available in many varieties from different suppliers, except for the static mount, thus a further selection of specific variants needs to be carried out in the next stage.

10.3.2 Movement of Part

This module covers both the movement of the part during cutting/welding, as well as the case where it only needs to be moved into position. The application scenarios show that the possible working principles of this particular module are many and that this module in many cases will be delivered by the customer. This is the case for the loading of the car body at Volvo and for the movement of the metal strip by the roll forming mill at IAI.

There is however still some cases where the movement or loading of a part needs to be delivered as a part of the RLC production unit, so for this reason some possible working principles for these cases are presented below:

Part Loading, *placing the part in position either automatically or manually*. Even if the part needs to be stationary during the cut/weld, it will still need to be placed underneath the scanner head. This can be done either manually, or with some autonomous mechanism that is integrated with the production lines. This could be something like a rotary table sketched in Figure 10.10(b).

Linear, *for simple movements of the part in a plane*. For the cases where the part needs to be moved in a plane, this can be done linearly with either an x - or xy -table, as

sketched in Figure 10.10(a). Alternatively, the rotary table in Figure 10.10(b) can also be used in this regard.

Robot, for complex movement of parts. This is assessed to rarely be the case for the cutting of parts, since it is easier and sufficient, in most cases, to only move the scanner head. However, for welding tasks a robot might be needed to accurately position parts together for welding. Another possibility is that a part needs to be moved from one station/conveyor to another by a robot, thus making it reasonable to perform the cut/weld operation as a step of the moving process, essentially saving a mechanism for moving the scanner head at a separate station.

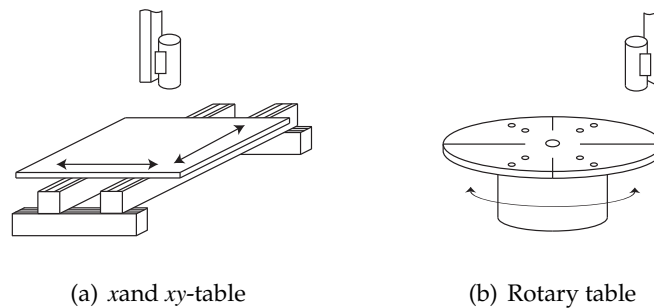


Figure 10.10: Linear working principles for moving the part.

Because there is a big diversity in the functional requirements for this module, a full search of possible working principles and their variants has not been carried out. In order to identify and fully define the working principles that should be offered for this module, a more thorough analysis and search would have to be carried out. Because of the limited timeframe of this thesis, this has not been possible. Instead, the three general working principles merely offers a suggestion to some possible working principles.

10.3.3 Laser Source, Cooling and Scanner Head

As it was described in Section 10.2.3, a total of three components are needed to create, focus, and position the laser beam for cutting or welding, together making up a single module.

The working principle for the laser has already been defined through the ROBOCUT project, as the only laser capable of meeting the beam quality requirements is a *single mode* (SM) fiber laser.

The scanner head is also defined through the ROBOCUT project. There is one variable however, because it is quite possible that there will be some difference

between a scanner head for cutting and welding respectively. The difference might just concern the DOE, described in Section 1.1, for the creation of the laser beam intensity pattern, however this is unknown at this point. Even though it is unknown, it is assumed that there will be some difference in the scanner head for cutting and welding applications, respectively, and thus they are defined as two distinct working principles.

Both the laser and the scanner head will need some form of cooling. In some cases the fiber lasers offer their own integrated cooling solution in the form of air conditioning [IPG Photonics Corporation, 2011], and in other cases the lasers rely on an external cooling system [Rofin-Sinar Laser GmbH, 2011]. Because it is not possible to find detailed information about laser cooling systems, this will not receive more attention. Instead it is assumed that the fiber laser is delivered with a suitable cooling system that has the capacity to cool the scanner head as well.

10.3.4 Controller and Software

The functionality of the controller and software module depends on the choices made for the movement of the scanner head and movement of the part. In terms of the controller there are basically three options:

PC Controller, *to control the scanner head and/or the linear modules.* The control of the scanner head can be handled by a pc controller with a user interface and control panel near the process. In case some additional equipment for moving the part and the scanner head has been chosen, these can also be controlled by this controller.

Robot Controller, *to control the robot.* When a robot is chosen, a robot controller is automatically needed to control it. In this case it is useful to integrate the control of the scanner head into the robot controller instead of using another controller. Doing so will ease the programming of the process, as it is then handled in one language and some issues concerning the synchronization between controllers and programs can also be avoided. In case some additional equipment needs to be controlled, special attention should be paid to the robot controllers capabilities to do so, as they have some limitations to the number of external axes that can be controlled.

Robot and PC Controller, *for additional monitoring and control.* Even though a robot controller is available, it could prove insufficient in some cases. For instance if the process needs monitoring with sensor readings, vision, etc. and/or a user control panel with process information and control. In this case, a PC controller could be used as a front-end controller to the robot controller. The

robot controller would still control the movements of the robot and the scanner head, however based on the commands coming from the PC controller.

Apart from the options above, it is possible that the customer has a controller or wishes another variety than what is available. For this reason the customer should also have the choice to deliver the controller.

The creation of programs for the controller based on the CAD-data can be handled through the customers own solution, either manually by an operator or through some third party CAD/CAM software. In case the customer does not have a satisfying solution, some additional pre-processing software can be delivered with the production unit:

CAD Add-in, *for an integrated solution.* It is assumed that most of the customers of the RLC production unit have a CAD system for designing the parts and the required cuts/welds. Hence, a solution, like the one presented in Chapter 7, with an add-in for the CAD system could be used. This would offer an integrated solution, as the designers would be able to both design and deploy the RLW/RLC process from the same CAD-system.

Stand-Alone, *for a separate high-level solution.* As there is a lot of different CAD-systems, it is unlikely that an add-in can be delivered for all of them. In such cases a stand-alone application is needed to program the cut/weld paths. This application will operate on a higher-level than the add-in as it will not include parts design, but will only concern the cut paths as it will operate with standard CAD formats.

10.4 Selecting and Evaluating modules

With the required modules and their working principles identified, some specific module variants can be selected based on an evaluation against the functional and technical requirements.

The selection is carried out by choosing a couple of suppliers for each of the modules, and then select the variants according to the identified working principles found in the previous section and the requirements that was found in Section 10.1.2. In reality, the selection would be carried out through acquiring offers from different suppliers, and then enter agreements with those that offer the best solutions.

At this point it should be mentioned that a selection of variants for the movement of part module is not carried out. This is because of the vast amount of possible working principles and varieties that exist for this module. Hence, the specific module variants should not be selected until the working principles of the module have been identified.

For reasons of reference, the suppliers of the module variants that have been used are listed in Table 10.5:

Suppliers of linear module for movement of scanner head:

Adept [Adept Technology, Inc., 2011]
Güdel [Güdel AG, 2011]
Schunck [Schunk GmbH & Co., 2011]

Suppliers of robots for movement of scanner head:

ABB Robotics [ABB Group, 2011]
KUKA Roboter [KUKA Roboter GmbH, 2011]
Fanuc Robotics [Fanuc Robotics GmbH, 2011]

Suppliers of fiber lasers:

IPG Photonics [IPG Photonics Corporation, 2011]
Rofin-Sinar Laser [Rofin-Sinar Laser GmbH, 2011]

Table 10.5: *The chosen suppliers of module variants.*

10.5 Embody Modular Architecture and Framework

With all of the module variants are selected, it quickly becomes difficult to keep track of all the possible combinations between module types and variants. For this reason the modules and their information needs to be structured to give an overview. This will be done by organizing the module variants in a *Product Variant Master* (PVM).

10.5.1 Product Variant Master

The PVM model, seen in Figure 10.11, consists of a part-of and a kind-of model, respectively. In the part-of structure, all of the modules and parts that makes up the product are listed. In case a module or part has different variants, this is modeled with a kind-of structure, which contains all the possible variants. Taking a bicycle as an example, the part-of structure would contain a frame, tires, gear, seat, etc. and the kind-of structure could contain different types of frames (e.g. standard, mountain, city), tire types and sizes, and so on.

As indicated on Figure 10.11, it is also possible to include relations between modules and parts. This is done by drawing the relation together with a description of the relation. In the bike example a possible relation could be that a standard frame puts a limit on the possible tire types and sizes.

Altogether, this provides an overview of the product structure, while also providing an overview of the variants for each module and their relations. For more detailed information about PVM, the reader is referred to Hvam et al. [2008].

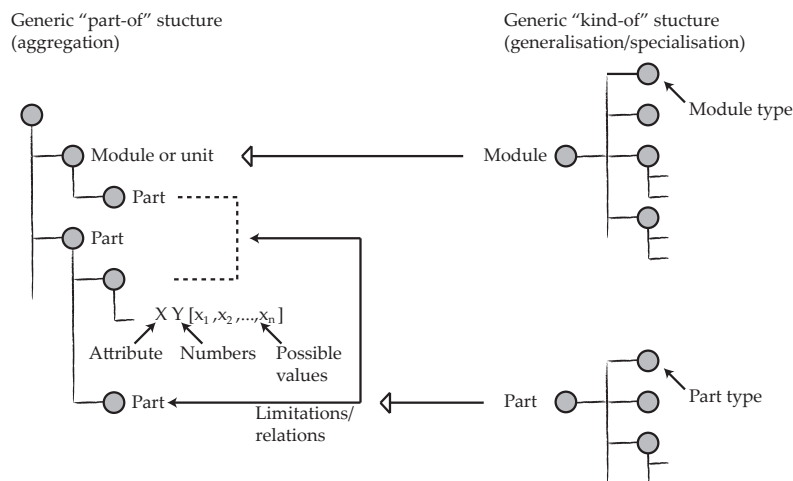


Figure 10.11: *The Product Variant Master [Hvam et al., 2008, p. 60].*

To construct the PVM the free application *Product Model Manager* (PMM) by Incore Systems Aps [2011] is used. Figure 10.12 shows a screen shot from the PMM application.

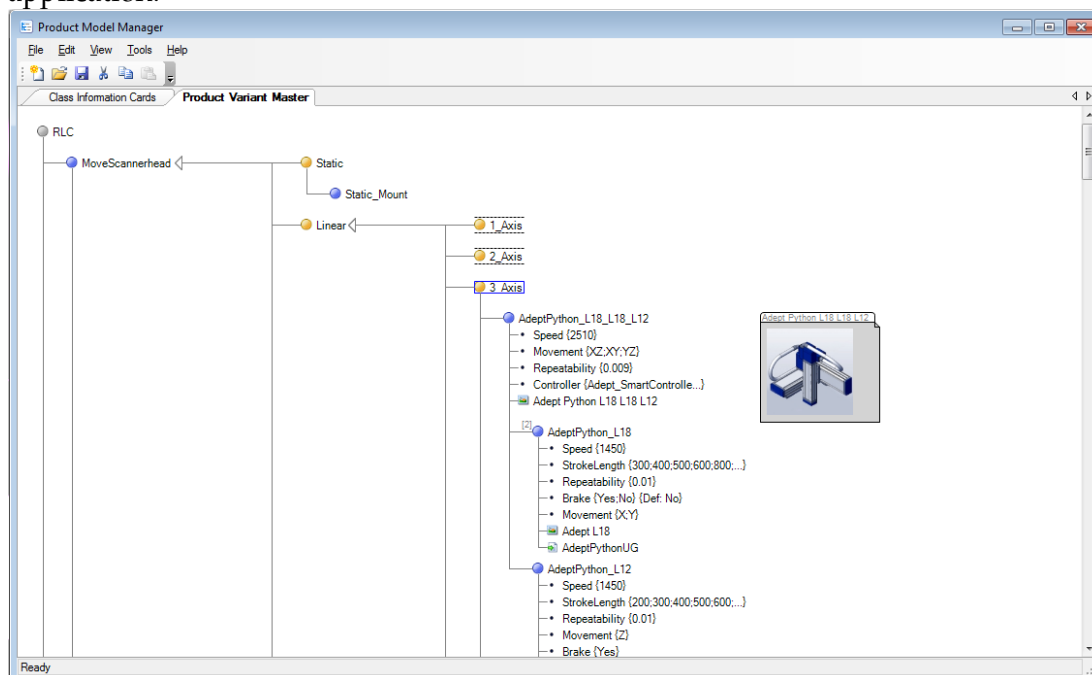


Figure 10.12: *Screenshot from the Product Model Manager application used to create the Product Variant Master.*

The PMM is a simple and easy to use application, however it is also quite limited. That is, it is not possible (at least not directly) to export the information and relations of the PVM to use with other software, e.g. as a back-end for an online configuration system. If this kind of functionality is needed, there are other commercially available applications like the *Configit Product Modeler* by Configit A/S [2011].

The full PVM model containing all of the selected modules can be found on the

enclosed CD together with a copy of the PMM application in the folder *Part II/Product Variant Master*.

10.5.2 Identify Configuration Relations

When all of the module variants have been selected, and the full range of possible configurations has been identified through the PVM model, the next step is to determine how a specific configuration is chosen based on requirements specified by the customer. Put in other words, it is necessary to determine which information is needed from the customer (i.e. questions to ask) in order to select a suitable configuration to match the customers needs and requirements.

There are some considerations to take into account when determining which information is needed, and especially in relation to how the information is obtained from the customer. First of all, it must be considered what information the customer is readily able to provide, which is directly related to the customer segment that is aimed for. Another consideration is the amount of information requested from the customer. Generally more information will yield a better opportunity to obtain a close match between the needs and requirements of the customer and the configured production unit. On the other hand the customer should not be overburdened with questions and information requests. To recap, there is a balance between making an exact match and requesting too much information.

With this in mind the information needed from the customers were identified. This was done by first investigating the attributes of the modules and variants, while trying to identify some common general attributes that could be used to distinguish and prioritize them. These general attributes were then translated into questions to be answered by the customer.

The result of this process is illustrated through Figure 10.13, where all of the identified questions are listed on the left side, *Customer Specified Requirements*, while the general attributes of the modules are listed on the right side, *Module Attributes*. As it can be observed, the questions have been bundled together according to the module and the attributes that they are primarily related to. In case the questions or attributes have some secondary relations, meaning that they have an impact on other modules or questions, this is shown by a dotted arrow.

Through the customer specified requirements (questions), the module attributes and the primary and secondary relations between them, respectively, provides a general framework for prioritizing and choosing the most suitable module variants for the customer. As Figure 10.13 only provides the broad overview, the following section will go through the basic idea behind choosing each of the modules based on the specified customer requirements.

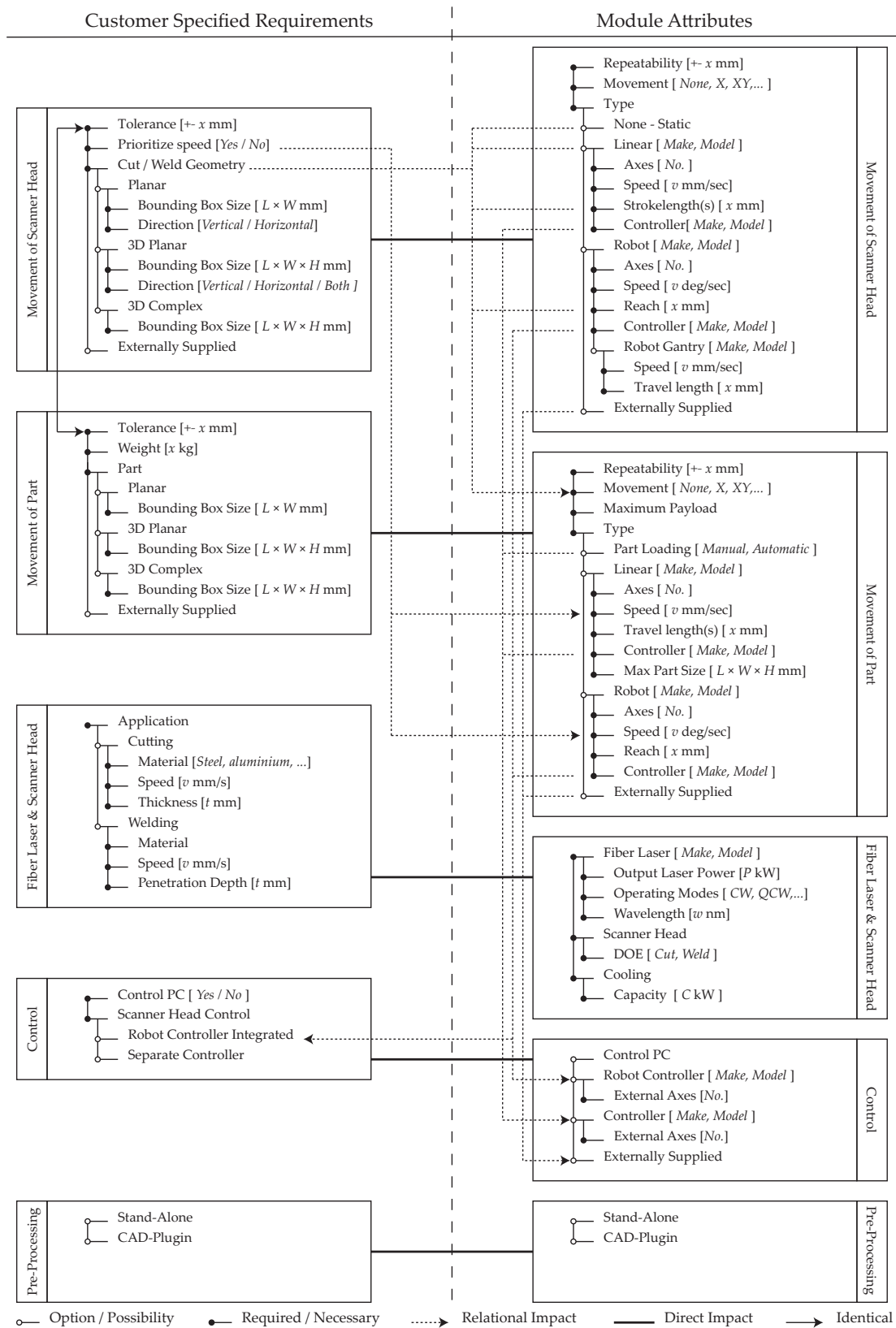


Figure 10.13: Configuration Relations.

10.5.3 Choosing the Movement of Scanner Head

The choice of module variant for the movement of scanner head depends mainly on the cut/weld shape size, and whether it is defined in planes or in more complex shapes. Note that the cut/weld shape is defined as the path in space that the laser beam needs to travel in order to complete the cut or weld.

As seen in Figure 10.13, there are basically four basic module variants for the movement of scanner head module: None, Linear, Robot and Externally supplied. The latter type is only applicable if the customer chooses to supply the movement of the scanner head. In order to determine which of these basic variants are most suitable, a simple classification of the possible cut/weld geometries has been made, illustrated by Figure 10.14.

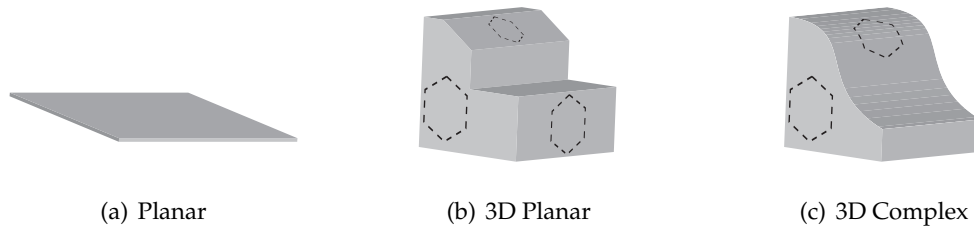


Figure 10.14: The three basic classifications of the cut/weld geometries. The dotted lines indicate example geometries/paths.

The idea behind this classification is that the customer chooses the classification that best fits the expected application scenario. Besides this, the customer needs to specify the *Bounding Box Size* of the shape, i.e. the size of the smallest box that can hold the complete shape inside. Based on these inputs and the working area of the scanner head, it is possible to estimate a suitable module variant.

Besides identifying the needed movement of the scanner head, it is also necessary to make sure that the movement is done within the tolerances specified by the customer. Note that the tolerances are defined by the summed tolerances of the scanner head, the movement of part module, as well as the part loading/movement module.

In case speed is a priority to the customer, this can be specified to suggest the fastest solution within the given specifications.

10.5.4 Choosing the Movement of Part

This choice depends highly on both on the cut/weld shape and the chosen module for movement of scanner head, as it is also indicated by the relation arrows on Figure 10.13. To further explain, consider the case where a robot has been chosen for the

movement of scanner head, but parts of the cut/weld shape is outside its reach. In this case the part might need to be moved, rotated or turned around. For the reasons discussed in Section 10.3.2, this is a very complex choice with a lot of if's and unknowns, and thus it has not been possible within the time frame of this thesis to identify an approach for carrying out this choice. Instead, the customer can specify if the part needs to be loaded or moved, and then assess and choose a suitable module variant.

Provided, that it was possible to suggest a suitable module variant, this would also have to take the weight and size of the part into account. The tolerance of the part movement should also be taken into account, as described in the previous section. Related to this, a tightly specified tolerance of the cut/weld, might make a movement of both the scanner head and the part impossible. In this case another solution needs to be found, where either the scanner head or the part moves. In the worst case scenario this could be an indication that the RLC production unit is not suitable for that particular application.

10.5.5 Choosing Laser source, Scanner Head and Cooling

As it was discussed in Section 10.3.3 it is only possible to create a laser beam of the required quality by using a single mode fiber laser. This simplifies the choice of the laser, as this reduces the problem to estimating the required size of the laser (i.e. required laser power). Also, recall that the cooling system is assumed to be delivered along with the fiber laser. The choice of scanner head is also simple, as it only depends on whether the customer needs cutting or welding of parts.

The estimation of the laser power depends on whether the customer needs cutting or welding. Depending on the application the laser power is estimated as described in the two following sections.

Estimating Required Laser Power for Cutting

The required power for laser cutting depends on a lot of different variables. For the case of conventional *melt and blow* laser cutting, the required laser power among others depend on the cut kerf width, speed, melting point, density, etc. [Steen and Mazumder, 2010, p. 157]. In case of remote laser cutting process, the required process will most likely depend on more variables, like the laser beam pattern for instance.

As there is no experimental data available on the performance of the remote laser cutting technology, a simplified model for the required laser power for conventional

laser cutting is used [Steen and Mazumder, 2010, pp. 157-158]:

$$E = \frac{P}{t \cdot v} \Leftrightarrow P = E \cdot t \cdot v \quad (10.1)$$

Where E is the *severance energy* in J/mm², P is the laser power in watts, t is the thickness in mm and v is the cutting speed in mm/sec.

The severance energy is determined first, based on the expected potential of the RLC technology. According to [The Danish National Advanced Technology Foundation, 2010, p. 3, Figure 2b] it is expected that the cutting of 1 mm thick stainless steel can be carried out by a 3000 watt single mode fiber laser at a rate of 90 m/min. Inserting this in function 10.1 yields:

$$E = \frac{3000}{1 \cdot 1500} = 2$$

This provides the necessary severance energy for stainless steel only. In order to estimate the needed power for other materials, an estimation of the severance energies for other materials is needed. To do this, it is assumed that the relation between severance energies for different materials are the same for remote laser cutting and conventional melt and blow cutting. Based on this assumption, the severance energies was estimated from the information about severance energies for different materials given in [Steen and Mazumder, 2010, p. 159]. The estimated severance energies are listed in Table 10.6.

Material:	Estimated E	Material:	Estimated E
Stainless Steel	2	Polystyrene	0.38
Mild Steel	1.54	Nylon	0.38
Titanium	2.14	ABS	0.38
Aluminum	2.14	Polycarbonate	0.38
Copper	4.6	PVC	0.3
Brass	3.4	Epoxy	0.54
Polyethylene	0.78	Glas	3.08
Polypropylene	0.46	Silica	18.46

Table 10.6: *The estimated severance energies for different materials.*

Based on the severance energies and equation 10.1 it is possible to estimate the required laser power, from information about material, thickness and required cutting speed. At this point it should be noted that the cutting speed will never be able to exceed 90 m/min, as this has been defined as the maximum obtainable moving speed for the cutting head [Olsen, 2011, p. 6]. As an example a customer could specify that a laser cutting of 1.5 mm aluminum plate at a rate of 50 m/min is needed, thus requiring

a laser power of:

$$P = 2.14 \cdot 1.5 \cdot 833 = 2674 \text{ W} \quad (10.2)$$

It must be emphasized that this is an estimation that is based on the simplified melt and blow model and the assumption that relations between severance energies of materials for conventional laser cutting. Once the RLC process has been completely developed, the actual performance and required laser power needs to be determined through experiments.

Estimating Required Laser Power for Welding

Just as it was the case for laser cutting, the required power for laser welding also depends on many different parameters including weld width, penetration depth, reflectivity, melting point, etc. [Steen and Mazumder, 2010, p. 211]. Again, as it was the case for cutting, it is possible that there are some additional variables pertaining to the new remote laser cutting technology.

Unfortunately there is no simplified model for estimating the required laser power for welding. Some detailed models does exist, but they require data that is not readily available for the customer [Steen and Mazumder, 2010, p. 211]. Instead, the estimation will be based on experimental data, as it will also be the case when the RLW technology has been developed. Experimental data for the welding of X100 steel with a fiber laser from in Quintino et al. [2007] is used. A graph representation of the date is shown in Figure 10.15.

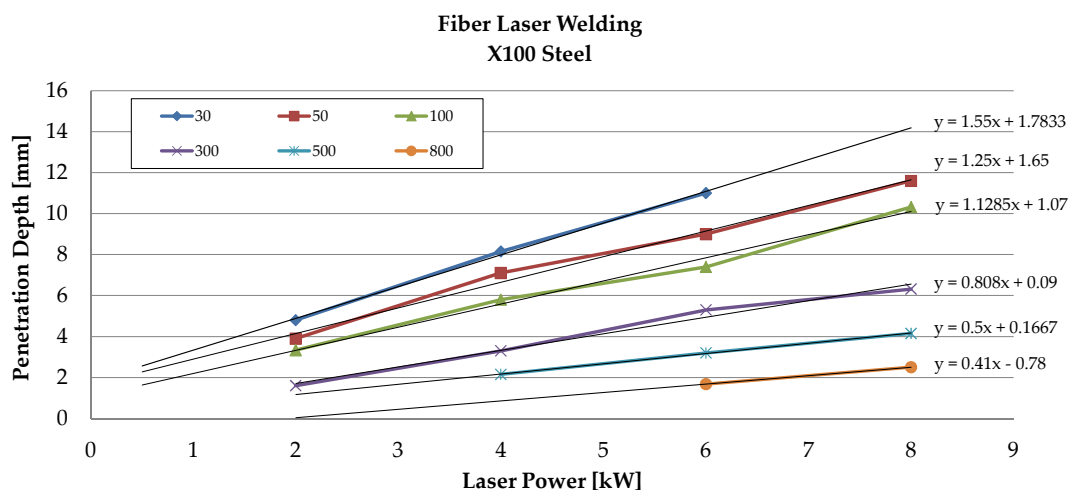


Figure 10.15: Penetration depth as a function of the laser power at different speeds (given in cm/min) for X100 steel [Quintino et al., 2007].

As it is seen from Figure 10.15 there is linear relation between the laser power and

the penetration depth for each of the different welding speeds. Hence, a linear fit is applied to each of the speeds, to extract the line equations. Unfortunately, there is not a clear tendency in the correlation between the slopes of the lines, making it impossible to carry out an interpolation for obtaining data at other weld speeds. For this reason, it is only possible estimate the required laser power for the speeds shown on Figure 10.15.

Again, it is emphasized that this is estimation cannot be considered as correct, but merely represents guess as well as a possible method for carrying out an estimation based on experimental data. Once the RLW technology has been completely developed, experiments should be conducted to obtain data (similar to what is used in this section) corresponding to the actual capabilities of the RLW process.

It was only possible to find data for fiber laser welding of X100 steel, and therefore the customer is not provided any other choices of material at this point.

10.5.6 Choosing Controller and Software

The choice of controller depends on the choices made for the movement of the scanner head and the part. If a robot has been chosen for either of these modules a controller is already included in the setup. In this case the customer can decide to integrate the control of the scanner head into the robot controller, or instead choose a PC controller to control the scanner head separately. As previously discussed, the customer could also choose to integrate the control of the scanner head into the robot controller, and then choose the PC controller for additional process monitoring and control (Section 10.3.4).

If no robot has been chosen, the customer has the choice between buying a PC-controller or supplying a controller for controlling the scanner head and possibly other equipment (i.e. if linear modules has been chosen).

In terms of the software this is a completely optional choice. This module is also not directly related to any other modules, except that the software is used to provide program code for the controller.

CHAPTER 11

Software Requirements

Software requirements specification defines what a software implementation is supposed to do before it is actually built. It may seem obvious to do this, but many projects are nevertheless delayed, because the development begins before the project team has clearly defined the requirements for the software.

When a project team starts on a new software project the spirits are often high. Programmers are often tempted to dive in and begin building the software when they have a feeling of what the software should do. And actually it is not very surprising: Programmers often have advanced programming skills without ever having to figure out and write down requirements in a systematic way. Programmers often develop their skills by building software that they intend to use themselves. In such case the programmer knows intuitively what the software is supposed to do from the beginning of the project.

Most software is however built to meet the needs of a customer or user of some sort. In order to fulfil the needs of the user the behaviour of the software must be planned before the software is built. This is where *Software Requirements Engineering* comes to play. Software Requirements Engineering deals with developing an accurate and complete definition of the functionalities and behaviour of the software that serves as the basis for software development [Greene and Stellman, 2005, Section 6.3].

Software Requirements Engineering often focuses on writing a Software Requirements Specification (SRS) document that contains the complete description of the behaviour of the software. For the development of the ROBOCUT configurator a separate SRS document will not be written as the software to develop is not that comprehensive. Instead this chapter contains the information that would normally be contained in an SRS document. The IEEE standard "Recommended Practice for Software Requirements Specifications" [IEEE, 1998] is used as a guide line. The standard describes the content and qualities of a good SRS and presents several sample SRS outlines. However it does not identify any specific method, nomenclature, or tool for preparing an SRS. The reader would benefit from looking in the standard before reading the rest of this chapter.

11.1 Scope

The software product to be produced is the "ROBOCUT Configurator". The ROBOCUT Configurator will enable users to configure a production unit for remote laser cutting using the ROBOCUT technology. The configurator will focus (as much as possible) on user requirements rather than letting the user select variants freely. The software should give proper recommendations if there exists a preferable choice for the user.

One of the main objectives is that the configurator is developed as a web site in order to reach as many users as possible.

The goal is not to make a system that works as a shop but instead provide the user with insight into the possibilities of remote laser cutting. Therefore the user will not submit actual orders but inquiries instead. It can be considered a non-binding inquiry for more information on the configured system.

The potential users of the ROBOCUT configurator are companies that are interesting in using remote laser cutting or welding. The configurator is not meant to be a tool for a "ROBOCUT sales assistant", but should be used directly by employees at the interested companies.

11.2 Product Perspective

This section shortly puts the ROBOCUT configurator into perspective with other related products.

A large number of configurators already exist. A common feature of these configurators is that they let users add and/or change functionalities of a core product. A good example is the Apple Stores Mac Pro Configurator, see Figure 11.1.

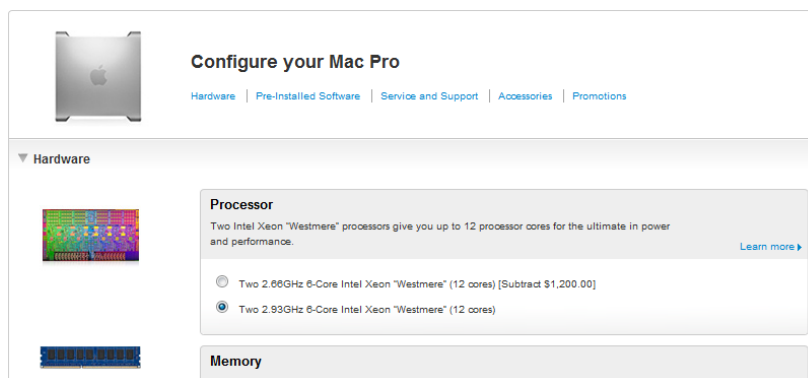


Figure 11.1: Apples Mac Pro configurator system, seen on 26/04/2011 at store.apple.com.

The ROBOCUTConfigurator will be different from these configurators in the sense

that it will hide modules that conflicts with the users requirements for the system. The ROBOCUT Configurator is also very different in the sense that it does not configure a product but a production system.

11.3 Specific Requirements

This section presents specific requirements for the software. They should reflect the needs of customers or users. As the ROBOCUT configurator is a research project the software is not ordered by a customer. For this reason the requirements cannot be developed in corporation with a customer as it would normally be the case.

The specific requirements are often classified into two or three categories. Based on the scope of the software to develop in this thesis the requirements are categorized into *functional requirements* and *non-functional requirements*. This section also contains design constraints that can be imposed by hardware limitations, other standards, etc.

The distinction between these categories of requirements may not be as clear cut as the definitions given below in the respective sub sections might suggest. The categories are just a mere tool of keeping requirements organized [Sommerville, 2006, p. 118].

Only a selection of the requirements is presented here. The rest can be found in Appendix E.

11.3.1 Functional Requirements

The functional requirements defines the fundamental actions that must take place in the software when processing the inputs and generating the outputs. The functional requirements can vary a lot in level of detail. It is important to avoid imprecision in the requirements as this will cause problems when developing the software [Greene and Stellman, 2005, Section 6.3.1.1].

The functional requirements can be classified into sub-functions or sub-processes [IEEE, 1998, p. 16]. This will not be done here as the number of requirements generally is fairly small.

Name	FR-1: Letting user specify requirements.
Summary	The user shall be able to define requirements for each module. The user specified requirements will then determine what variants are presented to the user.
Rationale	The user may not be aware of the variants available and what their abilities are. Therefore the configurator shall focus on the user requirements and determine the suitable variant(s) for the user.

Table 11.1: *Functional Requirement 1.*

Name	FR-2: Submitting an inquiry.
Summary	When the user has configured the desired system, the user should be able to submit an inquiry.
Rationale	The configurator should not be a sales tool but a tool for informing about the possibilities of using the ROBOCUT technology. Therefore the user cannot "buy" a configured system but instead send an inquiry. An inquiry is a request of more information on the configured system and is unbinding for the user.
Requirements	This requires an account system so the details can be coupled with each submitted inquiry.

Table 11.2: *Functional Requirement 2.*

11.3.2 Non-Functional Requirements

The non-functional requirements are not directly concerned with the specific functions in the software. It does not relate to what the software is supposed to do but instead *how* it does it. This can contain characteristics like how easy the software is to use, how quickly it executes, how reliable it is, error handling, etc. The non-functional requirements are often much less specific and can therefore be difficult to verify. Therefore they are often tested subjectively [Greene and Stellman, 2005, Section 6.3.1.2].

Name	NFR-1: Updating displayed variants in real time.
Summary	The displayed variants of a module are (as far as possible) updated continuously according to the requirements specified by the user.
Rationale	If the displayed variants are updated continuously it will enable the user to see the direct effect of changing a requirement.
Requirements	This will require the use of a technology that supports partial web page updates. Suitable technologies are AJAX, flash and others.

Table 11.3: *Non-Functional Requirement 1.*

Name	NFR-2: Always showing all selected variants.
Summary	It should always be clear to user what variants of what module variants have been selected.
Rationale	This requirement will ensure that the user always has an overview of the selected variants of modules. It will also help the user to see what variants of modules are still missing before the user can send an inquiry.

Table 11.4: *Non-Functional Requirement 2.*

11.3.3 Design Constraints

Design constraints refer to some limitation on the conditions under which the software is developed. Design constraints are often imposed by the technology used, available time to develop, overall budget and so on. The design constraint in Table 11.5 relates to the technology used. The available time and overall budget will not receive attention here as they are clearly defined by this project being a 10th semesters Master's Thesis.

Name	DC-1: Web Application Framework.
Summary	In order to convey the need for the configurator to function in an online environment with partial web site updates the web application framework ASP.NET 4.0 ¹ is used.
Rationale	ASP.NET 4.0 allows for quick web site development with integrated database support. It furthermore allows use of a full featured programming language such as C#. ASP.NET is widely used on the internet and seems to be the obvious choice.
Requirements	The development will be carried using Microsoft Visual Studio 2010. For database handling Microsoft SQL Server Management Studio 2008 will be used. A web host supporting ASP.NET 4.0 and SQL Server 2008 is also needed for publishing the web site online.

Table 11.5: *Design Constraint 1.*

11.4 Summary

The functional requirements, non-functional requirements and design constraints have been stated. The number of requirements are relatively few which is a consequence of the limited time frame and resources available. Nevertheless these requirements provide a good guideline for specifying the design in Chapter 12.

CHAPTER 12

Software Design Specification

The software development process covers the design and programming of the software. The chapter will not discuss the actual implementation (source code) but will instead serve as a Software Design Specification (SDS). The SDS is normally a separate document that describes and defines the software in order to meet the functional and non-functional requirements set in the Software Requirements Specification [Pressman, 2001, p. 358]. The software is then implemented based on the SDS. An SDS is especially important when managing large software development projects where many people are involved.

12.1 Architecture

Complex software systems are often decomposed into sub-systems. The process of identifying these sub-systems and establishing a framework for sub-system control and communication is known as architectural design [Sommerville, 2006, p. 242]. In other words the architecture of a software can be understood as the highest-level breakdown of a system into parts.

The choice of architecture is essential as it often determines the limits of the software in terms of functionality, usability, performance, reuse and so on [Sommerville, 2006, p. 242]. Each architecture has pros and cons and the difficult part is to choose the best architecture for the desired application. The risks of choosing a poor architecture includes unstable software, the system is unable to support future business requirements or it is difficult to deploy or manage in a production environment [Microsoft, 2009, p. 4].

Before developing the architecture of the software it is very important to understand architectural styles¹ and that they can be used to describe different aspects of the same system. Architectures can mainly be divided into two styles: Structure and deployment. A combination of structural and deployment architectural styles is often used when building public web application. The structural architecture separates the concerns while a specific deployment architecture might be needed for security

¹Architectural styles are sometimes referred to as architectural patterns.

reasons, e.g. 3-tier².

The focus will be on defining the structural architecture for the configurator. The deployment architecture is usually studied for larger projects only.

12.1.1 Structural Architecture

The online configurator is developed using ASP.NET 4.0 in C#. Even though the architecture of a software almost never can be limited to a single architectural style, an online software like this is often described using layers.

Layered architecture refers to partitioning the functionalities of the application into stacked groups. The online configurator will have a layered architecture as seen in Figure 12.1. The functionalities within each layer are related and a layer may only interact with layers directly above it or below it. The cross cutting functionalities however span the layers and includes authentication, validation, exception management, etc [Microsoft, 2009, p. 64].

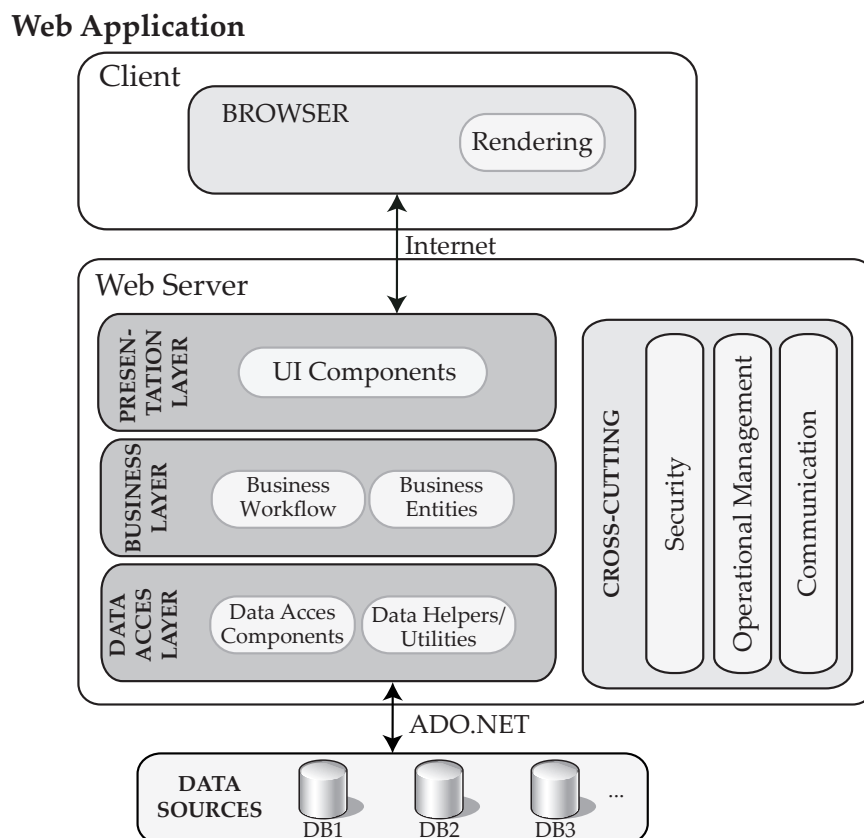


Figure 12.1: Structural architecture of online web configurator. With inspiration from [Microsoft, 2009, Figure 1, p. 278].

²A 3-tier application is an application program that is organized into three major parts, each of which is distributed to a different place or places in a network.

Remember that the layers of an application may reside on the same physical computer. Of course the client and web server are not the same physical computer, but the layers within the web server may all reside on the same physical computer. The layers are merely a way of organizing code.

The three layered architecture (Presentation Layer, Logic Layer and Data Access Layer) is a fairly generic way of describing the structural architecture of a web application. However it is important to keep this separation of code in mind when developing the application. Each layer has a responsibility [Microsoft, 2009, p. 58]:

Presentation layer contains the user oriented functionality. This is the layer that manages the user interaction with the system and is thus the bridge between the user and the logic layer.

Business layer implements the core functionality of the system. This is where information is handled and functions are called.

Data access layer provides the access to databases. The data access layer often consists of generic interfaces that the components in the business layer can use.

For each layer the configurator will have some specified characteristics.

Presentation Layer

For the presentation layer the configurator uses master pages to simplify development and to implement a consistent UI across all pages. This will drastically reduce development time.

For increased interactivity and background processing the web site will use AJAX³ technology. AJAX is a group of interrelated web development methods used on the client side that enables exchanging data asynchronously between browser and server to avoid full page reloads [Holzner, 2006, p. 1]. The asynchronous interaction between the presentation layer and business layer will also increase the responsiveness of the application and enable satisfaction of non-functional requirement 1, see Table 11.3, which concerns the updating of variants in real time.

Another important aspect of the presentation layer is user input validation. The application will use data validation techniques to protect the system from meaningless input.

³Acronym for Asynchronous JavaScript and XML.

Business and Data Access Layer

Because the configurator is a relatively small software project there is not much focus on separating the business and data access layer. However one very important aspect of the business layer is authorization. This is important for security and reliability. ASP.NET has built-in functionality for implementing security in web applications. This is important for satisfying non-functional requirement 6, see Table E.8, concerning a user account system.

Concerning the data access layer there is a number of technology considerations. The configurator will mostly only require support for basic queries⁴ and parameters, so the ADO.NET objects can be used directly. ADO.NET is a data access technology for the .NET framework that hides all underlying database code for the programmer. In some cases custom database code is necessary. The architecture of ADO.NET will not be presented here.

12.2 Database Design

The configurator reads data from a database as the modules and variants are presented to the user of the system. The structure of the data is a very important part of software design as it will have a huge influence on how the business and data access layer can be programmed.

The design process of the database is as follows (inspired by Microsoft [2011]):

- ◇ **Determine the purpose of the database.**
- ◇ **Find and organize the information required** - Gather all of the types of information that should be stored in the database.
- ◇ **Divide the information into tables and specify relationships** - Divide the information items into major entities or subjects. Each subject then becomes a table. Look at each table and decide how the data in one table is related to the data in other tables.
- ◇ **Decide what stored procedures are needed and how they should be constructed** - Stored procedures are often needed to simplify operations.

When the designing takes place it is an iterative process meaning that one may begin at step one again in order to refine the design once the last step has finished. In this thesis the design process is presented as one cycle.

⁴A database query is a piece of code (a query) that is sent to a database in order to get information back from the database. It is used as the way of retrieving the information from database.

12.2.1 Determine the Purpose of the Database

The database will be used for storing information about user accounts, inquiries and modules. ASP.NET has a membership feature that provides secure credential storage for application users. This membership feature is used, however some additional information than what is stored by the membership feature is needed like first name, last name, address, phone number, etc.

12.2.2 Find and Organize the Information Required

Table 12.1 and 12.2 shows the information needed about users and inquiries.

<hr/> UserId (identifies each user) Username Password E-mail address Name Address Phone number Position Company CVR-number <hr/>	<hr/> SystemID (identifies each inquiry) UserId All selected variants of modules Submit date and time <hr/>
--	---

Table 12.1: *Information needed about users.*

Table 12.2: *Information needed for each inquiry.*

Apart from these data the database should also contain details about all variants.

12.2.3 Divide the Information into Tables and Specify Relationships

The relationship between the tables can be stated in an entity relationship diagram (ERD) which is an abstract and conceptual representation of the data. Several notations for making such diagrams are available. Crow's Foot Notation will be used here. A short introduction to the symbols are presented, but for further information on the subject the reader is referred to [Halpin, 2000].

Figure 12.2 shows the relation between the most important tables concerning user accounts, inquiries and module variant information. The database diagram should be read top down and the `aspnet_Applications` table is at the top. This table enables use of the same database for several web sites as all user accounts will be associated with a specific application ID. In this case there is only one entry in the table as only one application is used.

A number of tables already exist as a part of the ASP.NET membership system. The

tables contain information about user name, password (encrypted), email, last login date and so on. Most central is the `aspnet_Users` table that contains the unique user ID and a user name. It is recommended not to edit or change the tables associated with the ASP.NET membership system, so in order to store additional data about the users the table `UserAddresses` is created. Furthermore a table named `Orders` will contain the inquiries from the customers.

The relationship between `aspnet_Applications` and `asp_Users` shows that an application can hold zero to many users. For each user there is an entry in the `UserAddresses` table that stores additional user information.

For each user there is zero to many orders (or inquiries) that each has a unique system ID. The `Orders` table contains the title of each variant of module that the customer has selected along with a submit date that tells when the inquiry has been submitted.

The tables within the blue area contains product information about all the different modules and variants available. These tables do not have any relation with the `Orders` table. The tables containing variant information should be used on the web site to display information to the customer about variants. There is no need for a relation between the `Orders` table and the tables containing variant information. The dashed lines indicates that the `Gantry` in `Orders` corresponds to an entry in the `Gantry` table. The reason behind the two dashed lines to the `Movement` table is that this table is used for both Movement of Scanner Head and Movement of Part.

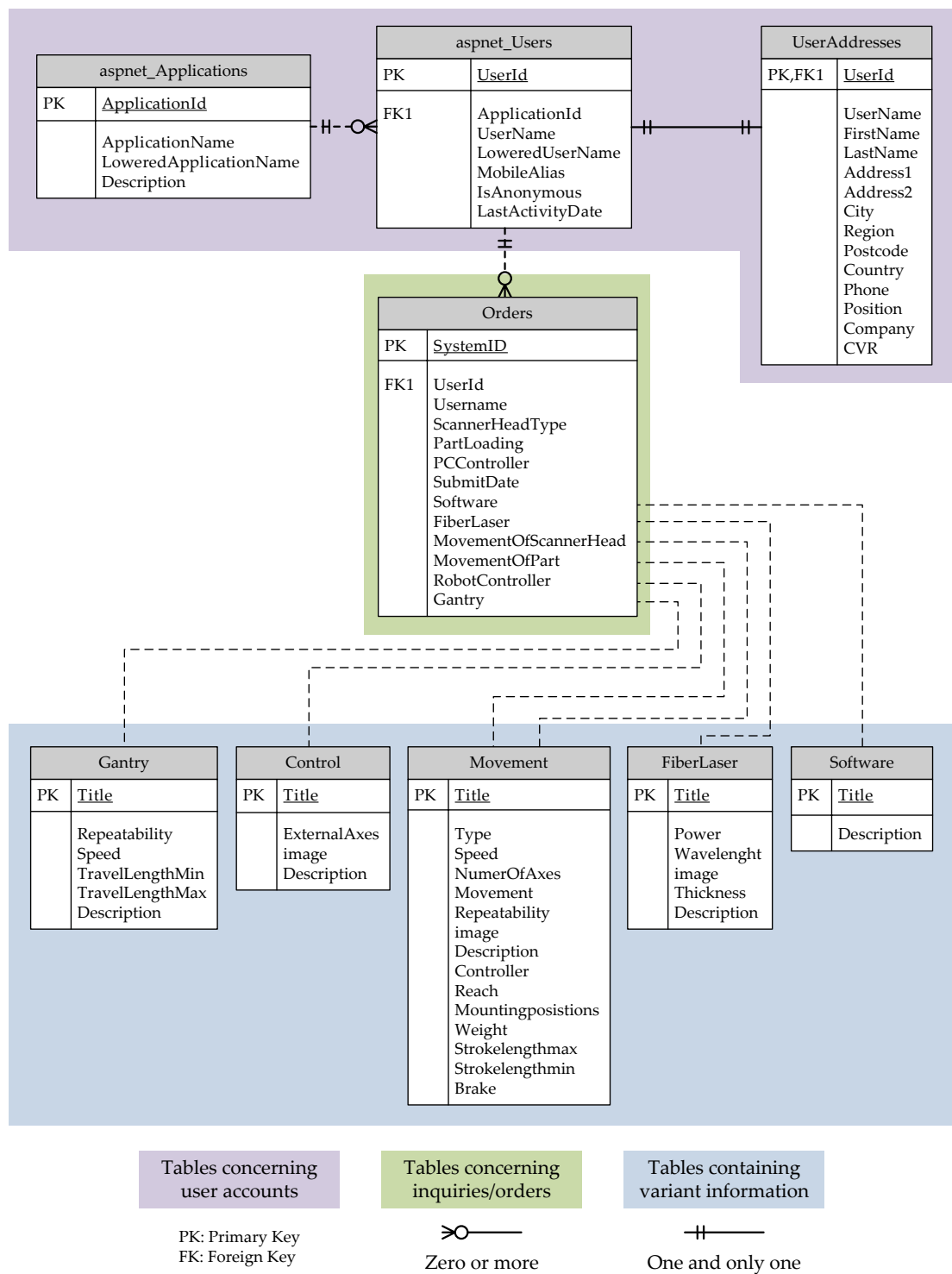


Figure 12.2: Entity relationship diagram (only showing selected tables).

12.3 User Interface Design

The user interface is probably the most important element of a computer-based system or product. If the interface is poor it could cause an otherwise well-designed application to fail completely.

Designing an appealing and functional user interface is difficult as it has as much to do with the study of people as with technology issues. Many considerations take place during the interface design and mainly three rules are kept in mind. These are also known as the "golden rules" [Pressman, 2001, p. 402]:

1. Place the user in control.
2. Reduce the user's memory load.
3. Make the interface consistent.

These rules receives attention throughout the user interface design. It is important to decide how the functionality should be split across independent pages. This is shown in a navigation map seen in Figure 12.3.

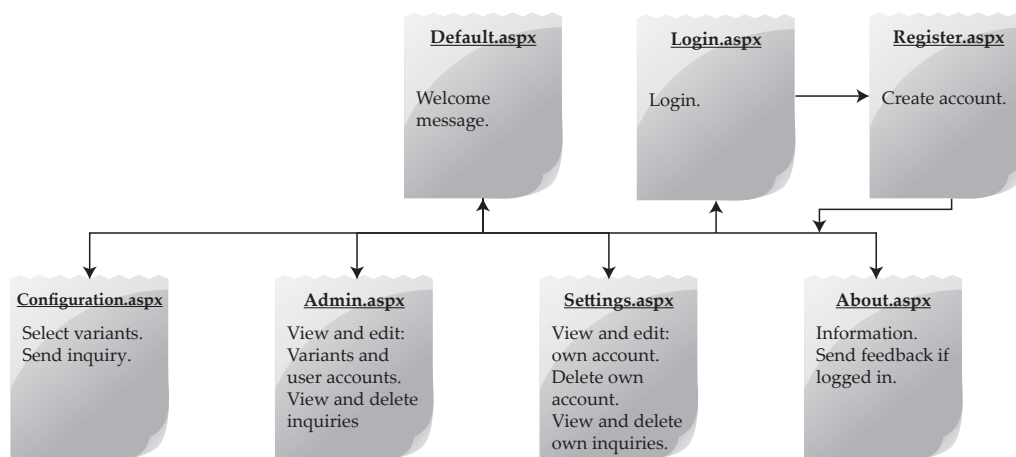


Figure 12.3: *Navigation map of web site.*

The navigation map shows the main functionality in each page and the navigation possibilities from the default page. The most important page is of course "Configuration.aspx" that contains the configurator. The navigation map shows that all pages except the registration page can be reached from the default page. The registration page can only be reached from the login page. The rest of this section focuses on the user interface design on the "Configuration" page.

The goal of the user interface design is to define a set of interface objects and their screen representations that enables a user to interact with the software in an

appropriate manner. Designing the interface is an iterative process that entails implementation and interface validation. In this process, sketches of the interface is made prior to the actual implementation. The final sketch for the user interface is depicted in Figure 12.4.

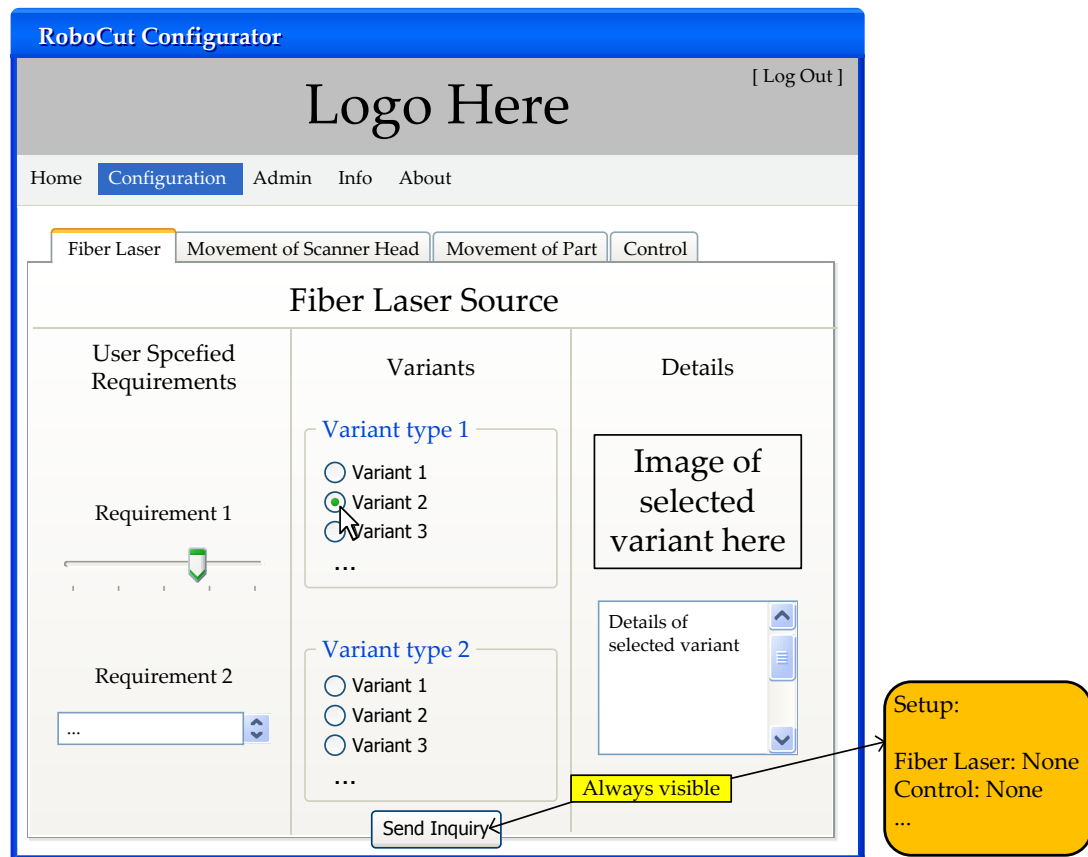


Figure 12.4: Final sketch for user interface.

Figure 12.4 shows the interface of the configuration page. The modules are placed as separate tabs in order to place the user in control. The user does not have to select the variants in any specific order. The body of each tab contains three columns: "User Specified Requirements", "Variants" and "Details". The system sorts and displays variants according to the user specified requirements. When the user selects a variant, the details of that variant will be displayed in the details column. In order to make the interface consistent, this basic layout with three columns is used for all four modules.

Reducing the user's memory load refers to the fact that the more a user has to remember, the more error-prone the interaction with the system will be [Pressman, 2001, p. 404]. To ensure that the user remembers what variants have been selected, a panel containing this information is present at the lower right corner of the screen while using the configurator. The panel also informs the user about the modules that still needs to be selected before an inquiry can be sent. Furthermore, a send inquiry

button will always be visible.

Another very important aspect of the user interface is to provide the user with help in an intuitive and easy way. Small question mark icons are placed around the website. Clicking on a question mark brings up a pop-up window that gives the user help and information.

CHAPTER 13

Verification and Validation

The developed configurator must be tested to ensure that it fulfils the requirements presented in Chapter 11. Verification and validation represents this task.

The terms verification and validation are often confused. In this thesis the following definition is used [Sommerville, 2006, p. 516]:

‘Validation: Are we building the right product?’
‘Verification: Are we building the product right?’

Verification and validation is performed using various testing techniques that depends on the nature of the software project. For the configurator, validation is carried out using *validation testing*, which should verify that the software built is what the customer ordered. This means testing that the software meets the requirements in Chapter 11.

The verification is carried out using *defect testing*, which should reveal defects in the system rather than simulating its operational use.

Before testing results are reviewed, Section 13.1 presents the user interface of the configurator.

13.1 Presentation of Configurator

Figure 13.1 shows a screen shot of the final user interface implemented according to the design specification in Section 12.3. The numbers in the figure are used to identify specific user interface components.

The reader is encouraged to visit the web site and try the configurator at <http://109717.testdom.dk/>. The web site is online at least until 21st of June 2011. The source code can be found in the folder *Part II/Configurator* on the enclosed CD.

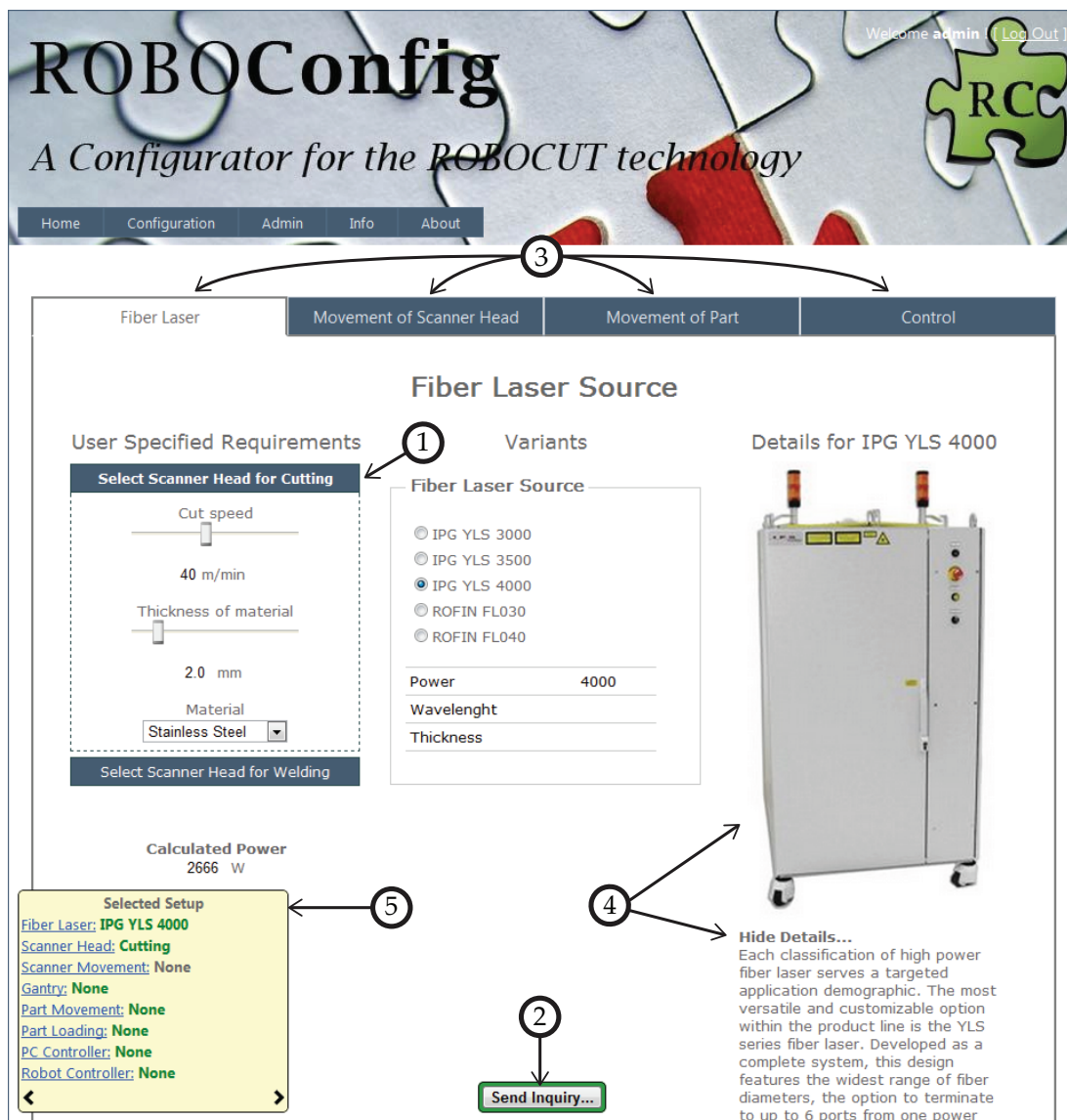


Figure 13.1: Screen shot of user interface (using Google Chrome 11.0.696.71).

Screen shots of all pages is shown in Appendix F.3 along with a brief description.

13.2 Validation Testing

The validation testing is carried out based on each requirement. Each test will be marked with a **Passed**, **Partly passed** or **Failed** as result. Some of the tests regarding non-functional requirements are based on a subjective assessment. The screen shot in section 13.1 is used as reference for some of the tests. The results of the validation testing are summarised in Table 13.1.

Requirement	Result	Summary
FR-1: Letting user specify requirements. Point 1 in user interface.	Passed	The user is able to specify requirements for all modules. However the user does not always get a chance to understand how the specified requirements affects the variants displayed. Thus, if the user has misunderstood a requirement the configurator will possibly display wrong variants to the user. This could be avoided by letting the user know how the specified requirements affects the displayed variants.
FR-2: Submitting an inquiry. Point 2 in user interface.	Passed	The user can submit an inquiry by pressing the button. The button is always visible and the user can therefore press the button before having acutally configured a system. In this case a pop-up shows the user what modules must be selected before an inquiry can be sent.
FR-3: Selecting variants in random order. Point 3 in user interface.	Passed	Using the tab panel, the user can switch between modules at any time. The tab panel will always save the state of each tab so that requirements does not have to be specified again when returning to a previously visited tab.
FR-4: View and edit user info and pending inquiries.	Partly Passed	The user is able to view and edit user account info. The user is able to delete inquiries if they are no longer of interest. However, it is not possible for the user to edit inquiries once they are submitted. If the user should be able to edit inquiries the system should check that the new variants chosen by the user are compatible. The administrator is able to modify all inquiries.
FR-5: Notifying user upon successful inquiry with inquiry ID.	Passed	Once an inquiry is successfully submitted an email is sent to the users email address containing the unique system ID. The system will warn the user if the mail was not successfully sent, in which case it is most likely to be caused by the user specifying an invalid email during the user account creation.

Table continued on next page

Requirement	Result	Summary
FR-6: Showing details of selected variant. Point 4 in user interface.	Passed	When the variant details are shown in the right column until the user selects another variant. This means that even though the user changes requirements leading to the selected variant not being displayed in the mid column, the right column will still display the same details.
NFR-1: Updating displayed variants in real time.	Passed	Asynchronous requests are sent to the server using AJAX as the user specifies requirements. The displayed variants are updated accordingly.
NFR-2: Always showing all selected variants. Point 5 in user interface.	Passed	A panel is displayed in the lower right corner of the screen containing all modules and selected variants. Because the panel may block the visibility of the actual site the user is able to move the panel to the left side of the screen as well.
NFR-3: Warn user immediately with informative text when selecting incompatible variant of module.	Partly Passed	The system will warn the user if an incompatible variant is selected. For instance, if the user selects an ABB controller for a KUKA robot. It was not possible however to incorporate all possibilities in the configurator. The user is for example able to choose a KUKA robot with a KUKA controller for movement of the scanner head while selecting an ABB robot for moving the part. It would be a smarter choice to select robots of same brand for compatibility.
NFR-4: Management of variants.	Passed	The administrator is fully able to add, edit and delete variants using the admin page.
NFR-5: Usability (by target user community).	Passed	Deciding whether or not the help and information present on the web site is adequate is difficult. Help and information is often one of the things that is mostly focused on during the software evolution process and it is thus passed for now.
NFR-6: Security.	Passed	Using the ASP.NET membership system means that the user account system is fully secure and sensitive information is stored safely.

Table continued on next page

Requirement	Result	Summary
NFR-7: Accessibility.	Passed	The accessibility of the configurator is very good. The website is online and all modern browsers are able to view and render the website.

Table 13.1: *Validation testing results.*

The results of the validation testing are overall positive. Not all tests were fully passed, but this is also due to the fact that a system like this should be able to evolve based on actual user feed back. This is also way the web site has incorporated a feed back function that allows a logged in user to submit feedback from the "About"-page.

The validation tests does overall show that the system conforms to the requirements in Chapter 11.

13.3 Defect Testing

For large software developments a test plan that draws up the approach for defect testing is defined. In the case of the online configurator this is not necessary because it is relatively easy to test most scenarios for defects without a systematic plan.

The defect testing is concerned with discovering undesirable system behaviour. This is why the test cases used are deliberately designed to expose defects, which is also why they often do not reflect how the system will normally be used [Sommerville, 2006, pp. 538-539].

It should be noted that the defect testing of the configurator actually consists of unit, integration and system testing. Unit testing is carried out while the system is developed and consists of testing individual units of code - preferably the smallest units of code possible. The integration testing occurs after unit testing and verifies that modules of code work together in the expected way. In system testing all software and hardware components are tested together. This just shows that testing is carried out during the development of the software.

The defects discovered in the defect testing process are shown in Table 13.2.

Title: Misses selection of scanner head	Severity: Low
Description: The type of scanner head is selected when the user presses "Select Scanner Head for Cutting" or "Select Scanner Head for Welding" in the requirements column in the Fiber Laser tab. Sometimes selecting of scanner head fails. This means that the user will have to press "Select Scanner Head for Cutting" or "Select Scanner Head for Welding" a second time in order for the scanner head to actually be selected.	
Title: Scroll position lost	Severity: Low
Description: The system updates the "Selected Setup" panel in the lower right corner every second in accordance with the variants the user has selected. If the user scrolls while the panel is updated the user may experience that the scroll position on the page "jumps" to another location on the page. This defect causes no problem but can be an annoyance for the user. This problem is caused by asynchronous update of the box showing selected variants every second.	
Title: Browser compatibility	Severity: Medium
Description: The cost of using new technologies is that out dated browsers will not be able render the web site properly. Several issues has been spotted with the layout and slider controls when using older versions of Microsoft Internet Explorer.	

Table 13.2: *Results of defect testing.*

Table 13.2 shows that not many defects were discovered in the defect testing process. Of course this does not mean that the software is free of defects or that it will behave as specified in all circumstances. A famous quote by Edsger Dijkstra says, that "program testing can be used to show the presence of bugs, but never to show their absence!" [Dijkstra et al., 1972, p. 6]. This quote very much states a crucial fact of software testing: Software could be testing for years and still not be 100% free of defects.

Fortunately the configurator is a relatively small software system and there is a limited amount of functions to test. Based on this, it is assessed that the system does not contain any severe bugs.

13.4 Summary

The conducted tests have shown that the software conforms the requirements specified in Chapter 11. The final acceptance tests are often carried out both by developers and the end user. The end user determines whether the software is ready for release. Unfortunately the time scope of the project did not allow for an end user test which is necessary before the software is ready for release. However it is advisable to revise the requirements and design specification before conducting end user tests.

CHAPTER 14

Discussion

A method for identifying the modular architecture of a RLC production unit was presented in Chapter 10. A method for identifying a modular architecture for a production system was not available. Therefore a modified version a method intended for the design of modular products was used. How suited the modified method is for identifying the modular architecture can be discussed. The contents of step 5, "Embody Modular Architecture and Framework", is in particular unclear. This was a contributing factor for the weak understanding of how customer specified requirements and modules could be coupled.

The fact that it was not possible to clearly identify the coupling between customer specified requirements and modules is reflected in the implementation of the configurator. The concept of the configurator is to display module variants based on user specified requirements. There are two approached for continuing the development of the configurator: Either the configurator should not display variants based on user requirements or the coupling between user requirements and modules should be studied further.

A way of avoiding displaying variants based on user requirements could be to let the user select a production scenario before moving to the actual configurator. Then only modules and variants applicable to that specific production scenario should be taken into consideration. In this way the configurator would also be more user friendly as some modules and options can be hidden.

No matter what approach is taken for further development of the configurator several issues need to be addressed. Especially these issues need attention:

- ◇ The capabilities and specifications of the ROBOCUT technology need to be fully defined.
- ◇ Getting experience with the RLC process in practice.
- ◇ A better understanding of the coupling between user specified requirements and feasible modules/variants is needed.
- ◇ More resources for programming. Preferably software engineers.

The capabilities and specifications of the ROBOCUT technology should be in place within a couple of years. The prospects of getting feedback from companies using RLC in real production scenarios are longer. It is however important for establishing knowledge about what production scenarios are best suited for the RLC process. This experience could be used to develop a more simple configurator as described above.

The configurator could also be directed against another group of users: Sales consultants. The sales consultants may use the configurator for advising customers over phone. This would again demand less from the configurator in terms of coupling between customer specified requirements and modules as the sales consultants should be trained in using the configurator.

The ROBOCUT configurator has shed some important light over what processes are important when modularising a production unit for configuration. The task remains to make more advances and sophisticated configurators while not sacrificing an intuitive user friendly experience. The ROBOCUT configurator might be so complex that more advanced technologies are needed in order to provide this. New technologies like 3D browser rendering might be necessary.

PART
III
CONCLUSION

CHAPTER 15

Conclusion

Scheduling of the laser cutting process

Part I of this thesis focused on scheduling the laser cutting problem. The work was based on a scenario at IAI concerning the use of RLC in a roll forming line. The scheduling problem was solved using two different approaches: Applying general theory of dispatching rules and combinatorial optimization. The two approaches solves the problem in two fundamentally different ways. Dispatching rules constructs a path by using a heuristic approach while combinatorial optimization searches for optimum solution.

The dispatching rules have a very low computation time and can handle very large problem instances. However it was difficult to construct the dispatching rules in such a way that it performs consistent with different parts and production parameters. In order to improve the performance composite dispatching rules were constructed. This resulted in a better performance but still not consistent.

Based on this, the dispatching rules alone cannot be considered a suitable method for scheduling a laser cutting job for this reason. There are however plenty of possibilities of combining dispatching rules with improvement heuristics or optimization techniques. Improvement heuristics are known to perform well for large problem instances and this could be an interesting field for further study.

A solver that finds the optimum path using a branch-and-bound method was successfully implemented. The computation time when using this approach increases exponentially with the problem size. As a result the maximum solvable problem size with this method was around 25 nodes. This is not sufficient since Ib Andresen have parts containing well over 250 holes. A way of using the branch-and-bound solver for larger problem sizes would be to divide the nodes into groups. The optimum path between the groups following by finding the optimum path within each group.

This concludes that both the dispatching and the combinatorial optimization approach can be used to schedule the laser cutting problem. Based on the work in this thesis the branch-and-bound method can be directly applied to small parts containing less than 25 holes.

An Add-in has been developed for the CAD software Autodesk Inventor that

enables a user to define cutting tasks, run the scheduler and view the resulting path. The Add-in serves to define scheduling problems in the form of tasks and constraints.

Development of a ROBOCUT Configurator

A methodology for designing modular products was modified by changing the scope to fit modularization of a production unit. This methodology was applied to the remote laser cutting production unit.

The output of the methodology was used as input for developing a configurator. The configurator focuses on translating user specified requirements into recommended variants of predefined modules. The relationship between requirements and variants identified by the methodology proved to be inadequate for developing the complete configurator. This is partly because the relationship between requirements and variants proved to be very complex due to the countless amount of application scenarios. It is also due to the fact that the methodology did not focus enough on this part. Creating new methodologies is an iterative process and it should be redefined to have more focus on identifying these complex relationships. Identifying these relationships is difficult because it has not been done before.

It is advised that any further work on the configuration of production units focuses on refining the methodology. This should include a better description of the relation between requirements and variants. This can be done by limiting the number of targeted application scenarios.

Another possibility is to change the concept of the configurator so it does not focus on user specified requirements. This could be done by presenting the user with a predefined set of application scenarios to choose from before using the configurator. The configurator would then be based on the chosen production scenario.

The idea of developing a configurator for a production system is new and interesting. This is why more research is needed on how to prepare production units for configuration. Configurators for production units will also benefit from advances in internet technology. Three dimensional rendering of modules could for example help build more intuitive configurators for complex systems.

Summary

This master thesis is a part of the ROBOCUT research project. The project's vision is to develop a revolutionary new laser cutting technology, ROBOCUT, which relies on a multi-beam principle, where the cutting process is performed with a complex laser beam pattern rather than a traditional single round laser beam.

The thesis focuses on two separate parts. Part I focuses on scheduling of the laser cutting process, while Part II defines a methodology for modularizing a production unit and preparing it for configuration.

Scheduling of the laser cutting process

The RLC problem is researched based on a roll formed part from Ib Andresen. A solution of the scheduling problem using the basic dispatching rules is attempted first, however as they are only able to construct a path in terms of a single objective they do not yield some feasible solutions. In an effort to improve the solutions, it is attempted to combine the basic dispatching rules into composite dispatching rules that are able to handle multiple objectives. Doing so does increase the performance of the obtained solutions, however the composite dispatching rules prove to be difficult to compose in a way that will yield consistent results across different problem instances. In addition these methods did not guarantee an optimal solution.

In order to achieve optimal solutions, a combinatorial optimization scheduler is explored and implemented next. The implementation uses branch and bound to find the optimal solutions. This can be done for up to 25 nodes with a computation time around 6 minutes. The solutions obtained through combinatorial optimization is then analyzed in terms of the fastest obtainable on-the-fly cutting of the example part. From this it is concluded that an on-the-fly cutting of the example part using the remote laser cutting can be carried out at a speed of around 200 mm/s.

To ease the use of the scheduler, an interface for the CAD application Autodesk Inventor 2011 is developed. This allows a user to easily define the cutting tasks for scheduling, then carry out the scheduling using the developed scheduler and then finally to get a visual presentation of the results. This demonstrates a possible solution to an integrated planning and scheduling system, using the existing applications that

the users are accustomed to.

Development of a ROBOCUT Configurator

With the overall objective of developing a configurator for the remote laser cutting and welding production unit, the second part starts with the identification of a modular architecture. As the idea of developing a configurator for a production has seemingly not been done before, a method for the identification and design of modular products is adapted to fit the modularization of a production unit. By applying this method a modular architecture is identified based on five possible application scenarios of the remote laser cutting and welding technology. The modular architecture is however lacking in terms of fully defining and limiting the number of possible combinations of modules.

Finally, the identified modular architecture is used as a basis for the development of an online configurator. The configurator allows a customer to specify the requirements for the process, and based on this the configurator presents a number of suitable solutions. In the end, the developed configurator is assessed to function well and according to specifications, however it still needs a further development.

Bibliography

ABB Group (2011, June 2nd). Abb robotics. <http://www.abb.com/product/us/9AAC910011.aspx>.

PDF on the CD: [ABB.com] Webpages.pdf.

Adept Technology, Inc. (2011, June 2nd). Adept pythin linear modules. <http://www.adept.com/products/robots/linear-modules/python/general>.

PDF on the CD: [Adept.com] Webpages.pdf.

Antonova, G., G. Gladush, A. Krasnyukov, F. Kosyrev, and N. Rodionov (2000). The mechanism of remote cutting of metals by CO₂-laser radiation. *High Temperature* 38, 477–482.

PDF on the CD: [Tahmouch et. al 1997] Cutting by high power laser at long distance.pdf.

Applegate, D. L., R. E. Bixby, V. Chvátal, and W. J. Cook (2006). *The Traveling Salesman Problem - A Computational Study*. Princeton University Press.

ISBN: 978-0-691-12993-8.

Autodesk Inc. (2011, February 14th). Autodesk - Developer Center - Autodesk Inventor. <http://usa.autodesk.com/adsk/servlet/index?siteID=123112&id=1079044>.

PDF on the CD: [Autodesk.com] DeveloperWebpages.pdf.

Belforte, D. A. (2010). The worst is over - industrial laser market recovers. *Industrial Laser Solutions for Manufacturing* 25.

PDF on the CD: [Belforte 2010] The Worst Is Over - Industrial Laser Marked Recovers.pdf.

Benhamou, F. and N. Jussien (2006). Trends in constraint programming.

PDF on the CD: [Benhamou 2006] Trends in constraint programming.pdf.

Bi, Z. M., S. Y. T. Lang, W. Shen, and L. Wang (2008). Reconfigurable manufacturing systems: the state of the art. *International Journal of Production Research* 46, 967–992.

PDF on the CD: [Bi 2008] Reconfigurable manufacturing systems - the state of the art.pdf.

-
- Chen, J. Y., M. E. Pfund, J. W. Fowler, D. C. Montgomery, and T. E. Callarmand (2010). Robust scaling parameters for composite dispatching rules. *IIE Transactions* 42, 842–853.
PDF on the CD: [Chen et al. 2010] Robust scaling parameters for composite dispatching rules.pdf.
- Configit A/S (2011, May 5th). Configit Webpage. <http://www.configit.com>.
PDF on the CD: [Configit.com] Webpages.pdf.
- Danmarks Statistik (2011, June 2nd). Gf2: Generel firmastatistik efter branche og enhed. <http://www.statistikbanken.dk/statbank5a/default.asp?w=1024>.
PDF on the CD: [Statistikbanken.dk] MetalcompaniesDK 00-08.xls.
- Davis, M. M. and J. Heineke (2005). *Operations Management: Integrating Manufacturing and Services*. McGraw-Hill Irwin.
ISBN-13: 978-0-07-111408-0.
- Dijkstra, E. W., O.-J. Dahl, and C. A. R. Hoare (1972). *Structured Programming*. Academic Press Inc. (London) Ltd.
ISBN: 0-12-200550-3.
- Edmonds, J. (1965). Paths, trees, and flowers. *Canadian Journal of Mathematics* 17.
PDF on the CD: [Edmond] Paths, Trees and Flowers.pdf.
- Fanuc Robotics GmbH (2011, June 2nd). Industrial robots. http://www.fanucrobotics.de/en/Products/A_Industrial-Robots.aspx.
PDF on the CD: [Fanucrobotics.de] Webpages.pdf.
- Federal Information Processing Standards (1993). Publication 183 - integration definition for function modeling (IDEF0). <http://www.idef.com/IDEF0.htm>
PDF on the CD: [FIPS 183] IDEF0.pdf.
- Greene, J. and A. Stellman (2005). *Applied Software Project Management*. O'Reilly Media, Inc.
ISBN: 0-596-00948-8.
- Grundfos A/S (2011, June 2nd). Fakta om grundfos koncernen. <http://www.grundfos.dk/web/homedk.nsf/CoverPages/About+us>.
PDF on the CD: [Grundfos.com] Webpages.pdf.
- Güdel AG (2011, June 2nd). Modules: Linear one- and multi-axis portals. <http://www.gudel.com/modules/>.
PDF on the CD: [Gudel.com] Webpages.pdf.
- Halmos, G. T. (Ed.) (2006). *Roll Forming Handbook*. CRC Press.
ISBN: 978-0-8247-9563.

-
- Halpin, T. (2000). Entity relationship modeling from an orm perspective: Part 3. *Journal of Conceptual Modeling* (13).
PDF on the CD: [Halpin 2000] Entity Relationship modeling from an ORM perspective Part 3.pdf.
- Hatwig, J., G. Reinhart, and M. Zaeh (2010). Automated task planning for industrial robots and laser scanners for remote laser beam welding and cutting. *Production Engineering* 4, 327–332.
PDF on the CD: [Hatwig et. al 2010] Automated task planning for robots and laser scanners.pdf.
- Holzner, S. (2006). *Ajax for Dummies*. Wiley Publishing, Inc.
ISBN-13: 978-0-471-78597-2.
- Hvam, L., N. H. Mortensen, and J. Riis (2008). *Product Customization*. Springer-Verlag.
ISBN: 978-3-540-71448-4.
- Ib Andresen Industri A/S (2011, February 18th). IAI's Webpage. <http://www.iai.dk>.
PDF on the CD: [IAI.dk] Webpages.pdf.
- IEEE (1998). IEEE Recommended Practice for Software Requirements Specifications.
PDF on the CD: [IEEE 1998] Recommended Practice for Software Requirements Specifications.pdf.
- Incore Systems Aps (2011, May 5th). Incore Systems Webpage. <http://www.incoresystems.dk>.
PDF on the CD: [Incoresystems.dk] Webpages.pdf.
- Ion, J. C. (2005). *Laser Processing of Engineering Materials*. Elsevier Butterworth-Heinemann.
ISBN: 0-7506-6079-1.
- IPG Photonics Corporation (2011). High Power Fiber Lasers for Industrial Applications. http://www.ipgphotonics.com/Collateral/Documents/English-US/HP_Brochure.pdf.
PDF on the CD: [IPG] HPLasersBrochure.pdf.
- IPU (2011, January 11th). Slides from: Robocut teknisk møde wp4.
- Jørgensen, S. N., K. Nielsen, and K. A. Jørgensen (2010). Reconfigurable manufacturing systems as an application of mass customisation.
PDF on the CD: [Joergensen et al 2010] Reconfigurable Manufacturing Systems as an Application of Mass Customisation.pdf.
- Kalpakjian, S. and S. R. Schmid (2006). *Manufacturing Engineering and Technology*. Pearson Prentice Hall.
ISBN: 0-13-148965-8.

-
- Kolakowska, E. (2010). *Integrated Planning and Scheduling in Fully Automatic Robot System*.
- Korte, B. and J. Vygen (2002). *Combinatorial Optimization - Theory and Algorithms*. Springer-Verlag.
ISBN: 3-540-43154-3.
- Kotthoff, L. (2010). Constraint solvers: An empirical evaluation of design decisions. *Computing Research Repository*.
PDF on the CD: [Kotthoff 2010] Constraint solvers: An empirical evaluation of design decisions.pdf.
- KUKA Roboter GmbH (2011, June 2nd). Kuka robotics. <http://www.kuka-robotics.com/en/>.
PDF on the CD: [KUKA.com] Webpages.pdf.
- Kuka Systems GmbH (2011, June 2nd). Laser remote welding. <http://www.kuka-systems.com/en/branches/technologies/Remote+Laser+welding/>.
PDF on the CD: [KUKA Systems] KS Roboscan brochure.pdf.
- Laporte, G. (1992). The traveling salesman problem: An overview of exact and approximate algorithms. *European Journal of Operational Research* 59, 231–247.
PDF on the CD: [Laporte 1992] The traveling salesman problem An overview of exact and approximate algorithms.pdf.
- Lawler, E. L. and D. E. Wood (1966). Branch-and-bound methods: A survey. *Operations Research Vol. 14*.
PDF on the CD: [Belforte 2010] The Worst Is Over - Industrial Laser Marked Recovers.pdf.
- Lee, Y. H., K. Bhaskaran, and M. Pinedo (1997). A heuristic to minimize the total weighted tardiness with sequence-dependent setups. *IIE Transactions* 29, 45–52.
PDF on the CD: [Lee et al. 1997] Heuristic to minimize total weighted tardiness with setups.pdf.
- MacDonald, M. and M. Szpuszta (2005). *Pro ASP.NET 2.0 in C# 2005*. Springer-Verlag New York.
ISBN-13: 978-1-59059-496-4.
- Mathias Theil Petersen (2011, February 21th). Recieved mail concerning parts and contact info.
PDF on the CD: [IAI 21-02-2011] Mail concerning parts.
- Merz, P. and B. Freisleben (2001). Memetic algorithms for the traveling salesman problem.
PDF on the CD: [Merz 2001] Memetic algorithms for the traveling salesman.pdf.

-
- Microsoft (2009). Microsoft application architecture guide.
ISBN: 9780735627109
PDF on the CD: [Microsoft 2009] Microsoft Application Architecture Guide.pdf.
- Microsoft (2011). Database design basics.
PDF on the CD: [Microsoft 2011] Database Design Basics.pdf.
- Nilsson, C. (2003). Heuristics for the traveling salesman problem. *Theoretica Computer Science Reports*.
PDF on the CD: [Nilsson 2003] Heuristics for the Traveling Salesman Problem.pdf.
- Olsen, F. O. (2011, April 11th). Optiksystem til robocut.
PDF on the CD: [Flemming Olsen 2011] Optiksystem til ROBOCUT.pdf.
- Ow, P. S. and T. E. Morton (1989). The single machine early/tardy problem. *Management Science* 35, 177–191.
PDF on the CD: [Peng and Morton] Single Machine Early-Tardy Problem.pdf.
- Pahl, G., W. Beitz, J. Feldhusen, and K.-H. Grote (2007). *Handbook of Constraint Programming - A Systematic Approach, 3rd ed.* Springer-Verlag London.
ISBN: 978-1-84628-318-5.
- Pinedo, M. and X. Chao (1999). *Operations Scheduling with Applications in Manufacturing and Services*. Irwin/McGraw-Hill.
ISBN: 0-07-289779-1.
- Pinedo, M. L. (2009). *Planning and Scheduling in Manufacturing and Services - Second Edition*. Springer Dordrecht Heidelberg London New Work.
ISBN: 978-1-4419-0909-1.
- Pop, P. C., O. Matei, and C. Sabo (2010). A new approach for solving the generalized traveling salesman problem. In *Hybrid Metaheuristics*, pp. 62–72.
PDF on the CD: [Pop 2010] A new approach for solving the generalized.pdf.
- Pressman, R. S. (2001). Software engineering: A practitioner's approach.
- Quintino, L., A. Costa, R. Miranda, D. Yapp, V. Kumar, and C. J. Kong (2007). Welding with high power fiber lasers - a preliminary study. *Materials and Design* (28), 1231–1237.
PDF on the CD: [Quintino et al 2007] Welding with high power fiber lasers.pdf.
- Reinhart, G., U. Munzert, and W. Vogl (2008). A programming system for robot-based remote-laser-welding with conventional optics. *CIRP Annals - Manufacturing Technology* 57(1), 37 – 40.
PDF on the CD: [Reinhart et. al 2008] Programming system for RWC.pdf.
- Rofin-Sinar Laser GmbH (2011). Operating Manual - Rofin FL 0XX Fiber Laser.

-
- Rosenkrantz, D. J., R. E. Stearns, and I. Philip M. Lewis (1977). An analysis of several heuristics for the traveling salesman problem. *Siam Journal on Computing* 6, 563–581.
PDF on the CD: [Rosenkrantz 1977] An analysis of several heuristics for the traveling salesman problem.pdf.
- Rossi, F., P. van Beek, and T. Walsh (2006). *Handbook of Constraint Programming*. Elsevier.
ISBN-13: 978-0-444-52726-4.
- Schrijver, A. On the history of combinatorial optimization (till 1960). *Discrete Optimization*.
PDF on the CD: [Schrijver] On the history of combinatorial optimization.pdf.
- Schrijver, A. (2003). Combinatorial optimization: polyhedra and efficiency.
PDF on the CD: [Schrijver 2003] Combinatorial optimization: polyhedra and efficiency.pdf.
- Schulte, C., G. Tack, and M. Z. Lagerkvist (2011). Modeling and Programming with Gecode.
PDF on the CD: [Schulte 2011] Modelling and programming with Gecode.
- Schunk GmbH & Co. (2011, June 2nd). Linear modules. <http://www.dk.schunk.com>.
PDF on the CD: [Schunk.com] Webpages.pdf.
- Sibbald, P. R., H. Sommerfeldt, and P. Argos (1992). Overseer: a nucleotide sequence searching tool. *Bioinformatics/computer Applications in The Biosciences* 8, 45–48.
- Sommerville, I. (2006). *Software Engineering (Eight Edition)*. Addison-Wesley Publishers.
ISBN: 7-111-19770-4.
- Srivastava, S., S. Kumar, R. C. Garg, and P. Sen (1969). Generalized traveling salesman problem through n sets of nodes. *CORS Journal* 7, 97–101.
PDF on the CD: [Srivastava et. al 1969] GTSP.pdf.
- Steen, W. M. and J. Mazumder (2010). *Laser Material Processing - Fourth Edition*. Springer-Verlag.
ISBN: 978-1-84996-061-8.
- Stefik, M. (1995). Introduction to knowledge systems.
ISBN: 1-55860-166-X.
- Stemmann, J. and R. Zunke (2006). Robot task planning for laser remote welding. In H.-D. Haasis, H. Kopfer, and J. Schönberger (Eds.), *Operations Research Proceedings 2005*, Volume 2005 of *Operations Research Proceedings*, pp. 729–734. Springer Berlin Heidelberg.
PDF on the CD: [Stemmann and Zunke 2006] Robot Task Planning for RLW.pdf.

-
- Stone, R. B., K. L. Wood, and R. H. Crawford (2000). A heuristic method for identifying modules for product architectures. *Design Studies* 21(1), 5 – 31.
PDF on the CD: [Stone et. al 2000] HMIMPA.
- Tahmouch, G., P. Meyrueis, and P. Grandjean (1997). Cutting by a high power laser at a long distance without an assist gas for dismantling. *Optics & Laser Technology* 29, 307–315.
PDF on the CD: [Tahmouch et. al 1997] Cutting by high power laser at long distance.pdf.
- The Danish National Advanced Technology Foundation (2010). Appendix 1: Project objective and content. <http://www.hoejteknologifonden.dk>.
PDF on the CD: [Højteknologifonden] Robocut Ansøgning Appendix 1.pdf.
- Ulrich, K. T. and S. D. Eppinger (2004). *Product Design and Development- Third Ed.* McGraw-Hill Irwin.
ISBN: 0-07-247147-8.
- Volvo Car Corporation (2011, January 11th). Moving variant hole cutting from body shop to final assembly.
- Zaeh, M., J. Moesl, J. Musiol, and F. Oefele (2010). Material processing with remote technology revolution or evolution? *Physics Procedia* 5(Part 1), 19 – 33.
PDF on the CD: [Zaeh et al. 2010] Material processing with remote technology - revolution or evolution.pdf.

PART
IV
APPENDIX

APPENDIX A

Ib Andresen Industri A/S

Ib Andresen Industri (IAI) is a danish family owned supplier company specializing in machining of steel and metals in mainly coils, sheets and tubes. The company dates back to 1967, where it was first founded by the civil engineer Ib Andresen. Since then the company has grown to be an international supplier company with departments in Denmark, Sweden, Norway, Hungary and Thailand, counting a total of 430 employees and an annual revenue (for 2010) of around 500 million DKK. [Ib Andresen Industri A/S, 2011].

The company has a corporate structure with five companies located in separate countries, as seen on Figure A.1, with the corporate headquarters located in Langeskov, Denmark. The company operates within four main business areas: Steel Service Center (Coil working, Sheets from coils and coil slitting), Sheet and plate working, Roll forming and Powder Coating. This project deals with the danish company, which is divided into five subdivisions in accordance with the four business areas, and specifically the project deals with the roll forming division as depicted by Figure A.1.

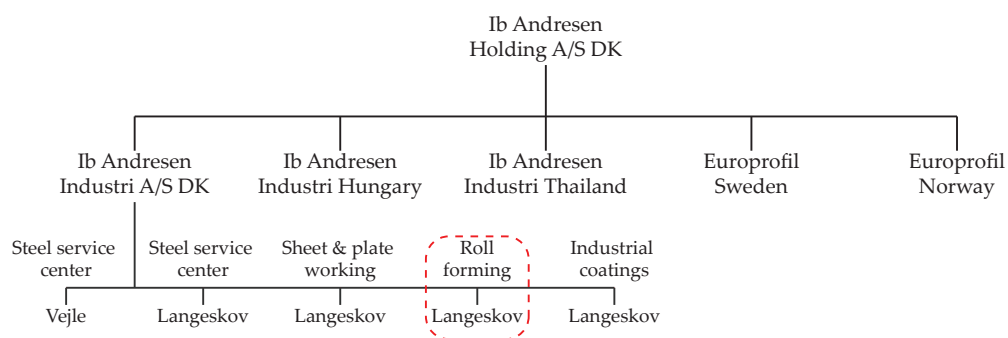


Figure A.1: The company structure of IAI, showing the five different companies and the five subdivisions of the danish company [Ib Andresen Industri A/S, 2011].

The roll forming division in Langeskov is participating in the development of the ROBOCUT technology with an implementation into their roll forming production lines in mind.

A.1 Roll forming

This section will briefly introduce the concept of roll forming and the roll forming process at Ib Andresen, before finally describing the potential problem areas of the roll forming process where the ROBOCUT technology could prove beneficial.

A.1.1 What is roll forming

Roll forming, also known as contour-roll forming and cold-roll forming, is a continuous forming process where a sheet metal strip is passed through a series of rolls, each performing a small step (bend, fold, embossing, etc.) towards a desired cross-section shape (profile), as illustrated by Figure A.2. The rolls are mechanically driven and placed in a *roll forming mill*.

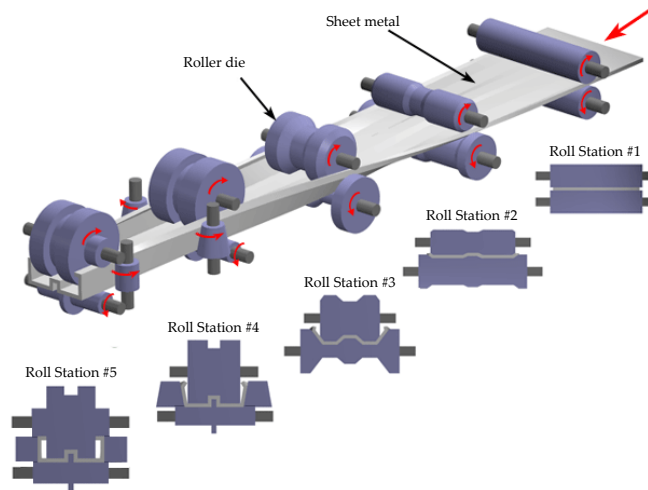


Figure A.2: Illustration showing the concept of roll forming. A sheet metal strip is fed through a series of consecutive rolls to produce the desired profile (Photo courtesy of CustomPartNet.com).

Roll forming can be used to produce a large variety of profiles with high complexity, including enclosed shapes. Some typical products produced by the process include tubes, frames for doors and windows, panels, business signs, girders, etc. The process is capable of handling sheet thicknesses ranging from 0.125 mm to 20 mm. The speed of the process varies according to the complexity and the sheet thickness, but they are generally below 90 m/min [Kalpakjian and Schmid, 2006, p. 448].

The process is typically used for producing large quantities and/or long length parts. The main reason being, that the setup of the process is expensive and time consuming. In addition, the rolls are expensive to manufacture and design. Designing and sequencing the rolls require a significant amount of experience, as a lot of different factors like springback, tolerances, straightness and flatness, surface appearance,

bending radius, number of necessary steps, etc. needs to be taken into account [Halmos, 2006, pp. 5-4..5-18]. These considerations usually results in a flower diagram as the one shown in Figure A.3, from which the contour rolls are then designed.

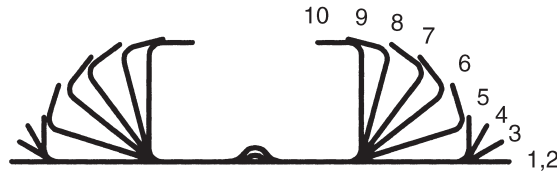


Figure A.3: A flower diagram showing the steps from metal strip (1) to the final profile (10) [Halmos, 2006, p. 5-4].

A.1.2 Roll forming at Ib Andresen

The profile rolling process at Ib Andresen Industri (IAI) generally consists of ten consecutive stations as depicted by Figure A.4. In what follows, numbers in braces refer to the station number in Figure A.4.

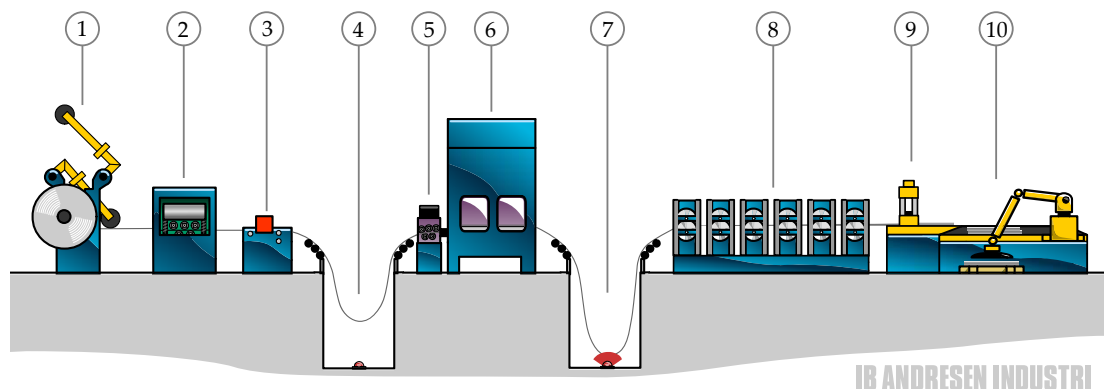


Figure A.4: The profile rolling process at Ib Andresen. Stations are numbered according to the processing sequence [Ib Andresen Industri A/S, 2011].

The roll forming process starts with the uncoiling of the metal strip to be roll formed. This is done by the *double uncoiler* (1), that allows for uncoiling of a coil, while another coil is loaded into the machine. Once a coil runs out, the double uncoiler switch around the coils and commences the uncoiling of the next coil.

After the strip leaves the coil it is put through the *flattener* (2), to remove the curvature from the strip. When a coil has run out, and the uncoiling of a new metal strip has commenced, the end of the metal strips are welded together at the *weld station* (3), to ensure a continuous flow. In case the metal strip needs to be pre-punched, marked, coined, notched, etc. it is carried out by the *press* (6). Since this operation requires that the metal strip is stagnant during the press action, it is fed by a *press feeder* (5).

To account for the disrupted flow caused by the press feeder and the press, the metal strip travels through a *free hanging loop with pit* (4)(7) before and after the press, thus acting like a material buffer. At this point the metal strip is ready to be roll formed, and it is passed through the roll forming mill (8), where it is gradually formed to the desired profile (Figure A.2), as described in the previous section.

Once the profile exits the roll forming mill, it is cut into pieces of the desired length in the *cut off press* (9). To ensure the continuous flow, this is a flying die press, meaning that the press follows the speed of the profile while performing the press action. After the profiles has been cut into length they roll onto the *run out table* (10), where they are handled and packed.

The roll forming division of IAI has 20 roll forming lines that follow the general process described above, however they vary in size and configurations. They are able to handle sheet metal thicknesses between 0.25 mm and 7.0 mm, widths ranging from 20 mm to 1000 mm, and lengths of the rolled parts between 150 mm and 16.000 mm [Ib Andresen Industri A/S, 2011]. The parts are typically rolled at a speed of 60 m/min, however the speed is typically reduced to 25 – 30 m/min for parts with complex holes (see Section A.8).

Parts are usually produced in batches, running for 1 – 1¹/₂ weeks at a time. This is due to the significant changeover time between different parts, which typically takes between one and two working shifts (i.e. 8 - 16 hrs.).

A.2 ROBOCUT potential in the roll forming process

The parts produced by IAI using the roll forming process often have several holes of varying size, location and shape. A typical part produced by IAI is seen in Figure A.5.

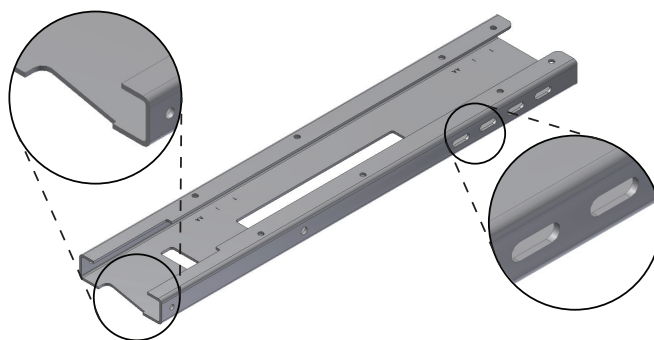


Figure A.5: A typical part produced by IAI using the roll forming process.

With the current roll forming process, all the holes of a part should ideally be pre-punched by the press ahead of the roll forming mill. Unfortunately there is some

problems associated with the pre-punching of holes:

Pre-punching some holes may cause problems, because they may cause defects and failures or even make the roll forming process impossible to carry out. There are several limiting factors to the successful pre-punching of holes, including hole size, shape and placement [Halmos, 2006, pp. 4-40]. The mechanical properties, the quality, and the thickness of the sheet metal are important factors as well [Halmos, 2006, pp. 10-33,11-3]. As an example, consider the part shown in - Figure A.5, that is currently produced by pre-punching all of the holes. According to IAI, the pre-punching of the slotted holes in the side and the notch in the end presents a challenge with unwanted deformations.

Pre-punched holes may deform, as they pass through the roll forming mill [Halmos, 2006, p. 9-14]. The forming of the sheet metal often causes both longitudinal and transversal distortions as well as changes in thickness, especially around bend lines [Halmos, 2006, pp. 11-1..11-19]. As a result, a pre-punched hole needs to be corrected according to the expected distortions when pre-punched (see Figure A.6), making it difficult to maintain a reasonable tolerance of the holes in the final parts.

It is difficult and expensive to post-cut holes, because this requires expensive tooling and/or additional handling and processing. In case a hole is impossible to pre-punch, there is basically two options for making the holes after the roll forming mill. It is possible to develop some flying die cutting tools just before, or as a part of, the cut-off press. This however is an expensive solution, as the tools needs to be specially developed for every part. As another option the holes needs to be cut in a separate process (e.g. by CO₂ laser cutting), requiring additional handling and an increased lead time.

Press die tool limitations, determines the number of different shaped holes, that can be created to a part. In some cases the number of different shaped holes in a part might exceed what is possible to implement in a single die tool. In this case the number of different shaped holes must be reduced, or the part may need to be produced on a roll forming line with a larger press capable of producing the holes.

Given the expected capabilities of the remote laser cutting technology developed through the ROBOCUT project, it could offer a solution, or improvement, to the problems just described. As mentioned in Section 1.1 the ROBOCUT project aims at developing a laser cutting head capable of laser cutting remotely without the use of assist gas. Because the laser cutting will be done remotely, the laser cutting head is able to cut the holes from a static position. Because of this, the laser cutting head can be conveniently placed both before and after the roll forming mill, but also in between

the rolls in the mill. In addition, the cutting head could be moved around the rolled part after it exits the roll forming mill, enabling it to cut holes into the part from all sides "on the fly". This is also possible with today's laser technology, however at lower speeds than the remote laser cutting technology is estimated to be capable of.

Thus the remote laser cutting head being developed by the ROBOCUT project has the potential to solve the problems of pre-punching holes by cutting the holes during or after the roll forming. At the same time, the difficulties regarding the deformation of holes through the roll forming mill are removed, as these holes can be cut after roll forming. Also, the limitations regarding the press die tool could be removed.

In addition to solving these problems, the ROBOCUT technology might also offer other improvements like:

Increased process speed, because the need for pre-punching the holes using a press is reduced, or even replaced by the remote laser cutting technology. Currently the speed of the roll forming process is somewhat limited to that of the press.

Increased flexibility, since the laser cutting head could not only remove the previous press die tool limitations, but also give the potential for a higher degree of customization of the holes from part to part during production.

Lowered cost, because the remote laser cutting of holes after the roll forming mill removes the current need for expensive tooling and/or extra processing and handling.

A.3 Visits at Ib Andresen

The project group has worked together with Ib Andresen Industries (IAI) during the project. This chapter will provide short summaries of the meetings with IAI. Reading the summaries should give the reader a better understanding of how issues have been discussed with the company and how helpful IAI have been.

Location: Ib Andresen Industries, Langeskov

Attending:

- ◇ Supervisor Morten Kristiansen
- ◇ Dan Gadensgaard
- ◇ Jonathan Skovhus
- ◇ PhD Student Martin Andersen

- ◇ Project manager at IAI Lars Hoffmann Pedersen
- ◇ Mechanical Engineering Student Matthias

Several topics were covered during the meeting and this chapter will roughly divide them into sections.

A.4 Problems associated with punched holes

Several characteristics of the current production at IAI were discussed at the meeting. Punching holes in the profiles before rolling can lead to problems if the holes are too close to edges. The material tends to buckle around the hole. Another issue is that holes punched before rolling tend to be more or less distorted after rolling. In order to counteract this effect, holes are punched so that after the distortion caused by rolling, the hole will get the desired geometry. Figure A.6 shows an example of a geometry punched that will be square after rolling.

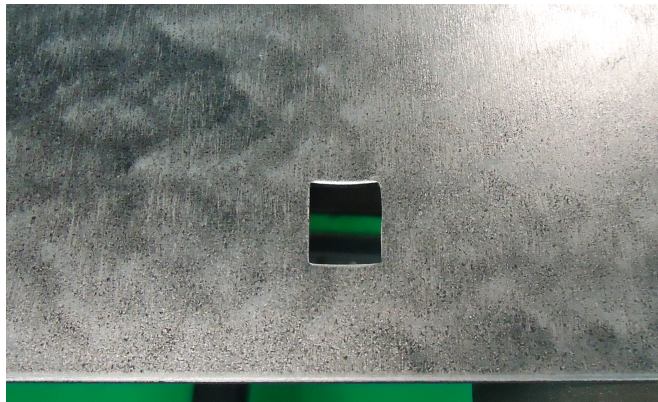


Figure A.6: Punched hole before rolling. The hole will be square after rolling.

A.5 CAD data

Part geometries are modeled using Autodesk - AutoCAD Inventor. It has been arranged that three part drawings will be handed to the project group along with information on what production lines are used for producing the products. Some floor plans will also be handed to the project group.

A.6 Changeover

Due to long and expensive setup times the production lines will produce the product for at least one to one and a half week at a time. A changeover will take about one whole shift (eight hours).

It is estimated that a changeover time of one to one and a half hour is acceptable for the laser cutter.

A.7 Software for defining holes

It is a problem that operators can not change program code. For example operators rely on technicians for calibrating the placement of a punched hole. It would be possible to make an interface for the machines with a GUI that would make it possible for the operators the take care of small calibrations. This task

A.8 Controlling the laser cutting process

The speed of the profile can vary if holes in the profiles meet several mills at the same time. How much the speed varies exactly is not known. This could be investigated further since it is relevant as to whether the laser cutting process needs to take these speed fluctuations into account.

Is has been suggested that the position of the profile along the mills could be controlled using reference holes. The holes could be detected be a laser or a vision system. At the moment the cutter has a precision of about 0.4 mm using a wheel to measure the position of the profile.

The speed of the profiles are up to 60 m/min. The typical speed for profiles with complex holes are 25 – 30 m/min.

A.9 Setup of Laser

The rolling mills are divided into cassettes which can be moved individually. It has been suggested that the laser cutting process could be dedicated into a singe cassette. In this way the laser process could be moved to another location if wanted. There could however be issues with safety.

APPENDIX B

Scheduling Constraints

B.1 Description of Constraints

Knowing and understanding the constraints are utmost important for actually modeling the RLC process for scheduling. The constraints will depend on many different process characteristics, that changes on according to the type of setup and application scenario as previously described. Therefore the constraints will be described along with a description of the scenarios in which they are relevant.

B.1.1 Cut geometries inside closed geometries first

Figure B.1 shows a piece of sheet metal (bordered with the dashed line) with cutting lines. The final part is shown to the right.

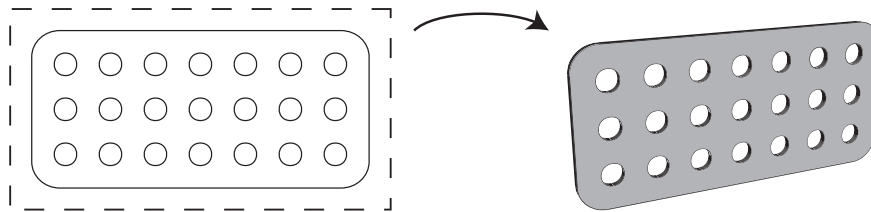


Figure B.1: *Illustration of cutting lines for cutting a plate with holes.*

In this example it is clear that the holes should be cut before the actual part is cut free. This is to ensure that the sheet is fixed while the holes are cut. This process constraint is independent of the setup and should therefore always be considered.

B.1.2 Minimize the process time

This is an important and always applicable constraint regarding a minimisation of the process time.

This constraint of course relies on a number of other constraints like how a job and task is defined. It is not possible to go further into details with this constraint now as it is also a matter of what it is possible to actually model and implement. Remember that

these constraint only serve as a basis for actually modelling the process for constraint optimisation.

B.1.3 Controlling the maximum/minimum cutting angle

When a geometry is laser cut remotely there will most likely be a cutting angle in the material because the cutting head is not right above the cutting point. This is illustrated in Figure B.2.

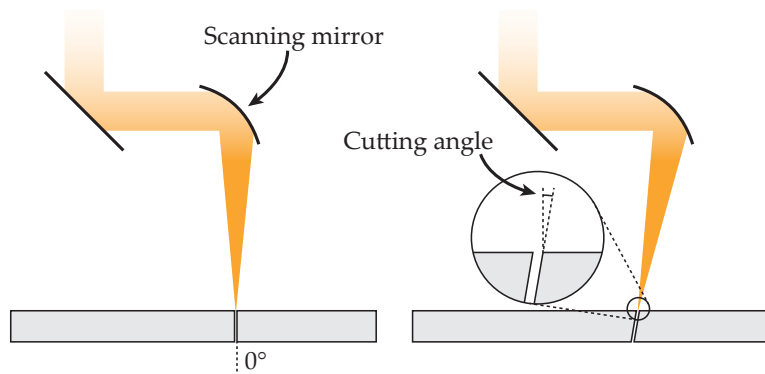


Figure B.2: *Illustration of a straight and angled cut.*

The cutting angle is dependent on the placement of the cutting head relative to the cutting point. This means that it is possible to somewhat control the cutting angle when it is possible to control the cutting head position relative to the cutting point. This can be achieved by having a movable cutting head or movable part.

B.1.4 Controlling the difference in cutting angles

If the cutting head or part moves while cutting a closed geometry (e.g. a circle) the cutting angle in the material when the cutting starts and ends might not be the same. This could be a problem when cutting in thick sheet metal as Figure B.3 illustrates. The red faces illustrate the cutting line.

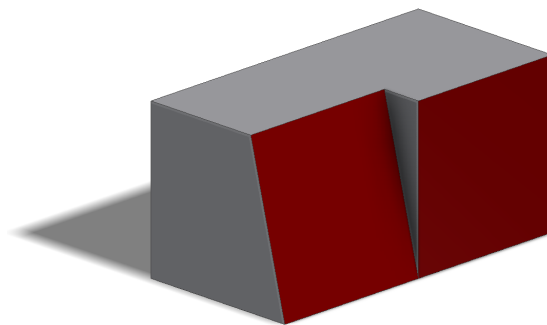


Figure B.3: *Illustration of a difference in cutting angle.*

If cutting a closed geometry the cut out may not actually be cut free and breaking the cut out free may result in burrs. This constraint can be both a soft and hard constraint. It is a hard constraint if a difference in cutting angles are not allowed and a soft constraint if there is a preferred difference in cutting angles.

This constraint is only relevant when it is possible to control either the position of the cutting head or part.

B.1.5 Satisfy limits of cutting head speed and acceleration

The cutting head can only move with a certain speed and acceleration that limits e.g. how the cutting angles can be minimized. This is clearly a hard constraint that must be satisfied. Of course it is only relevant if the cutting head is not stationary.

B.1.6 Cut geometries within the given time window

When dealing with cutting parts that are continuously moving there is only a certain amount of time to cut a geometry before it is too late. This introduces a *time window* for each geometry to be cut.

This process constraint is relevant when roll forming parts where material is continuously fed to the roll mills, see section A.1 about roll forming.

B.1.7 Overview and classification of constraints

It is important to distinguish between *hard constraints* and *soft constraints*. Hard constraints (as the name imply) are to be necessarily satisfied, while soft constraints only express a preference of some solution. Soft constraints are often modelled using cost functions. Table B.1 provides an overview and classification of the constraints.

Constraint:	Hard	Soft
Cut geometries inside closed geometries first	✓	
Minimize the process time		✓
Controlling the maximum/minimum cutting angle	✓	
Controlling the difference in cutting angles	✓	✓
Head and/or part speed and acceleration	✓	
Cut geometries within time window	✓	
Cutting holes before roll forming	✓	

Table B.1: Overview of the constraints with classification of hard and soft constraints.

APPENDIX C

Gecode Source Code

```
1 #include <gecode/driver.hh>
2 #include <gecode/int.hh>
3 #include <gecode/minimodel.hh>
4 #include <gecode/graph.hh>
5 #include <gecode/search.hh>
6
7 #include <algorithm>
8 #include <cmath>
9 #include <iostream> // I/O
10 #include <fstream> // file I/O
11 #include <iomanip> // format manipulation
12 #include <string>
13 #include "io.h"
14 #include "windows.h"
15
16 using namespace Gecode;
17 using namespace std;
18
19 /* ***** USE FOR READING COORDINATES FROM FILE ***** */
20 const char* coordinates = "coordinates.txt";
21 const char* constants = "constants.txt";
22
23
24 // Initialize row and column counters.
25 int RowNumCoordinates = 0;
26 int ColNumCoordinates = 0;
27
28 int RowNumConstants = 0;
29 int ColNumConstants = 0;
30
31 // Perform reading of data file into 2D array.
32 float** coordinatesArray = ReadTable(coordinates, RowNumCoordinates, ColNumCoordinates);
33 float** constantsArray = ReadTable(constants, RowNumConstants, ColNumConstants);
34
35 /* ***** */
36
37 const int PA_n = RowNumCoordinates; //Number of points
38 const int* PA_d = getDistance(coordinatesArray, PA_n); //Calculate distances
39 //const float* angle = getAngle(coordinatesArray, constantsArray, PA_n); //Calculate angles
40
41 const int cuttingarea = 400; //The cutting area is 500 [mm] i the x-direction.
42 const int delay = 100; //How far the part is from the due date line in [mm].
43 const int vskaere = constantsArray[0][0];
44 const int vflyt = constantsArray[1][0];
45 const int vconveyor = constantsArray[2][0];
46 const int ymin = constantsArray[3][0];
47 const int ymax = constantsArray[4][0];
```

```

48 bool multipleresults = false;
49
50 /// Problem instance
51 namespace {
52     class Problem
53     {
54     private:
55         const int _n; //< Size
56         const int* _d; //< Distances
57     public:
58
59         Problem(const int n, const int* d) : _n(n), _d(d) /// Initialize problem instance
60         {
61         }
62
63         int size(void) const /// Return size of instance
64         {
65             return _n;
66         }
67
68         int d(int i, int j) const /// Return distance between node \a i and \a j
69         {
70             return _d[i*_n+j];
71         }
72
73         const int* d(void) const /// Return distances
74         {
75             return _d;
76         }
77
78         int max(void) const /// Return estimate for maximal cost of a path
79         {
80             int m=0;
81             for (int i=_n*_n; i--;)
82                 m = std::max(m,_d[i]);
83             return m*_n;
84         }
85     };
86
87     Problem PA(PA_n,PA_d);
88
89 }
90
91
92 class TSP : public MinimizeScript
93 {
94 protected:
95     Problem p; /// Problem instance to be solved
96
97     IntVarArray succ; /// Successor edges
98
99     IntVar total; /// Total cost of travel
100
101     IntVar totalNoEndPoint; //The distance traveled when not moving back to point 0
102
103     IntVarArray time; //Array storing when each point is finished
104
105     IntVarArray timebegin; //Array storing earliest start time for each geometry

```

```

106
107 public:
108     /// Actual model
109     TSP(const SizeOptions& opt): p(PA), succ(*this, p.size(), 0, p.size()-1), total(*this, 0, p.max()), totalNoE
110     {
111         rel(*this, totalNoEndPoint == total - p.d(p.size()-1,0));
112
113         int n = p.size();
114
115         IntArgs c(n*n, p.d()); // Cost matrix
116
117         for (int i=n; i--; )
118             for (int j=n; j--; )
119                 if (p.d(i,j) == 0)
120                     rel(*this, succ[i], IRT_NQ, j); //This relation states that if two points have same coordinates
121
122
123         IntVarArgs costs(*this, n, Int::Limits::min, Int::Limits::max); // Cost of each edge
124
125         // Enforce that the successors yield a tour with appropriate costs
126         circuit(*this, c, succ, costs, total, opt.icl());
127
128
129
130
131         //circuit(home, c, x, y, z);
132         //x: are constrained to the values forming the circuit
133         //y: defines the cost of the edge for each node
134         //z: defines the total cost of the edges in the circuit
135
136         rel(*this, succ[n-1] == 0); //Forces route to last point in coordinate set before going back to point 0
137
138         // First enumerate cost values, prefer those that maximize cost reduction
139         branch(*this, costs, INT_VAR_REGRET_MAX_MAX, INT_VAL_SPLIT_MIN);
140
141         // Then fix the remaining successors
142         branch(*this, succ, INT_VAR_MIN_MIN, INT_VAL_MIN);
143
144         branch(*this, &TSP::post);
145     }
146
147     void more(void)
148     {
149         int i = 0;
150         int timetemp;
151         int duedate;
152         int releasedate;
153         do
154         {
155             if (i < 1)
156             {
157                 timetemp = coordinatesArray[0][4]/vskaere*1000;
158                 //cout << "\tTime(" << i << "): " << temp << " [ms]";
159                 rel(*this, time[i] == timetemp);
160             }
161             else
162             {
163                 timetemp += coordinatesArray[i][4]*1000/vskaere + p.d(i-1,succ[i-1].val())*1000/vflyt;

```

```

164         //cout << "\tTime(" << i << "): " << temp << " [ms]";
165         rel(*this, time[i] == timetemp);
166     }
167     i=succ[i].val();
168 }
169 while (i != 0);
170 //cout << "\t" << time << endl;
171 for (i=0; i<p.size(); i++)
172 {
173     duedate = coordinatesArray[i][0]*1000/vconveyor + delay*1000/vconveyor; // Due date [ms] = (x coordinate [mm])/v
174
175     releasedate = duedate - cuttingarea*1000/vconveyor + (coordinatesArray[i][1]-coordinatesArray[i][0])*1000/vconveyor;
176
177     rel(*this, time[i] < duedate);
178     rel(*this, time[i] - coordinatesArray[i][4]*1000/vskaere > releasedate); //For the release date the cutting time
179 }
180 //print(cout);
181 }
182
183 static void post(Space& Home)
184 {
185     static_cast<TSP&>(Home).more();
186 }
187
188 virtual IntVar cost(void) const /// Return solution cost
189 {
190     return totalNoEndPoint;
191 }
192 /// Constructor for cloning la s
193 TSP(bool share, TSP& s) : MinimizeScript(share,s), p(s.p)
194 {
195     succ.update(*this, share, s.succ);
196     total.update(*this, share, s.total);
197     totalNoEndPoint.update(*this, share, s.totalNoEndPoint);
198     time.update(*this, share, s.time);
199 }
200 /// Copy during cloning
201 virtual Space*
202     copy(bool share)
203 {
204     return new TSP(share,*this);
205 }
206
207 /// Print solution
208 virtual void
209     print(std::ostream& os) const
210 {
211
212     bool assigned = true;
213     for (int i = 0; i < succ.size(); i++)
214     {
215         if (!succ[i].assigned())
216         {
217             assigned = false;
218             break;
219         }
220     }
221     if (assigned)

```

```

222     {
223
224         if (!multipleresults)
225         {
226             clearTextFile("result.txt");
227         }
228
229         os << "\tTour:_ " << endl;
230
231         int i = 0;
232         do
233         {
234             os << (i == 0?"\t ":"") << i << (i == p.size()-1?"\n": "_->_");
235             writeAdd("result.txt", GetStrVal(i));
236             i=succ[i].val();
237         }
238         while (i != 0);
239
240         writeAdd("result.txt", "Cost:_ " + GetStrVal(totalNoEndPoint.val()) + "\n");
241         os << "\tTotal_distance_moved:\t" << totalNoEndPoint << "_[mm]" << endl;
242
243         i = 0;
244         do
245         {
246             int starttime = time[i].val()-coordinatesArray[i][4]*1000/vskaere; //Remember that the time once a
247             int duedate = floor(coordinatesArray[i][0]/vconveyor*1000) + delay*1000/vconveyor;
248             int releasedate = duedate - cuttingarea*1000/vconveyor + (coordinatesArray[i][1]-coordinatesArray[i
249
250             os << "\tCT(" << i << "):_" << starttime << ",_" << time[i] << "_[ms]" << "\t\tDD:_ " << duedate <<
251             writeAdd("resultextended.txt", GetStrVal(i) + "\t" + GetStrVal(starttime) + "\t" + GetStrVal(time[i]
252
253             i=succ[i].val();
254
255         }
256         while (i != 0);
257
258         writeAdd("resultextended.txt", "Cost:_ " + GetStrVal(totalNoEndPoint.val()) + "\n");
259
260         os << endl;
261     }
262     else
263     {
264
265     }
266 }
267 };
268
269 int
270 main(int argc, char* argv[])
271 {
272     clearTextFile("result.txt");
273     clearTextFile("resultextended.txt");
274     writeAdd("resultextended.txt", "Point\tStart\tFinish\tDD:\tRD");
275
276     cout << "Number_of_points:_ " << PA.size() << endl;
277     cout << "Coordinates:_ " << endl
278     << "\txmin\txmax\ty\tz\tCut_Length" << endl;
279

```

```

280     PrintMatrix(coordinatesArray, RowNumCoordinates, ColNumCoordinates); //Prints out the loaded coordinates
281
282     cout << "Constants:_ " << endl
283           << "V_skaere_[mm/s]:" << vskaere << endl
284           << "V_flyt_[mm/s]:" << vflyt << endl
285           << "Conveyor_speed_[mm/s]:" << vconveyor << endl << endl;
286
287     int numberOfSolutions = 0;
288     string input = "";
289     while (true) {
290         cout << "Number_of_solutions_to_find_(0=_best_solution):_";
291         getline(cin, input);
292
293         // This code converts from string to int safely.
294         stringstream myStream(input);
295         if (myStream >> numberOfSolutions)
296             break;
297         cout << "Invalid_number,_please_try_again" << endl;
298     }
299
300     cout << endl;
301
302     if (numberOfSolutions != 1)
303     {
304         while (true)
305         {
306             cout << "Would_you_like_to_save_multiple_solutions_to_\"results.txt\"?_Y/N:_";
307             getline(cin, input);
308
309             if (input == "Y" || input == "y")
310             {
311                 multipleresults = true;
312                 break;
313             }
314             else if (input == "N" || input == "n")
315             {
316                 multipleresults = false;
317                 break;
318             }
319             else
320             {
321                 cout << "Invalid_input,_try_again" << endl;
322             }
323         }
324     }
325
326     cout << endl;
327
328     SizeOptions opt("TSP");
329     opt.solutions(numberOfSolutions);
330
331     opt.icl(ICL_DOM);
332     opt.parse(argc, argv);
333     //opt.time(30*1000); //Time limit for search in [ms]
334     //opt.mode(SM_GIST); //Start gist
335
336     if (opt.size() >= PA_n)
337     {

```

```

338         std::cerr << "Error:_size_must_be_between_0_and_"
339         << PA_n-1 << std::endl;
340         return 1;
341     }
342
343     MinimizeScript::run<TSP,BAB,SizeOptions>(opt);
344     cout << endl << endl;
345
346     while (true) {
347         cout << "Scheduling_finished._Would_you_like_to_run_\" Placement_of_Scanner_Head_Script\"?_Y/N:_";
348         getline(cin, input);
349
350         if (input == "Y" || input == "y")
351         {
352             system("Placescannerhead.exe");
353             break;
354         }
355         else if (input == "N" || input == "n")
356         {
357             cout << "Done" << endl;
358             break;
359         }
360         else
361         {
362             cout << "Invalid_input_-_try_again" << endl;
363         }
364     }
365
366     return 0;
367 }
368
369 // STATISTICS: example-any

```


APPENDIX **D**
Functional Structure Diagrams

D.1 RLC ahead of or inside the roll forming mill at IAI

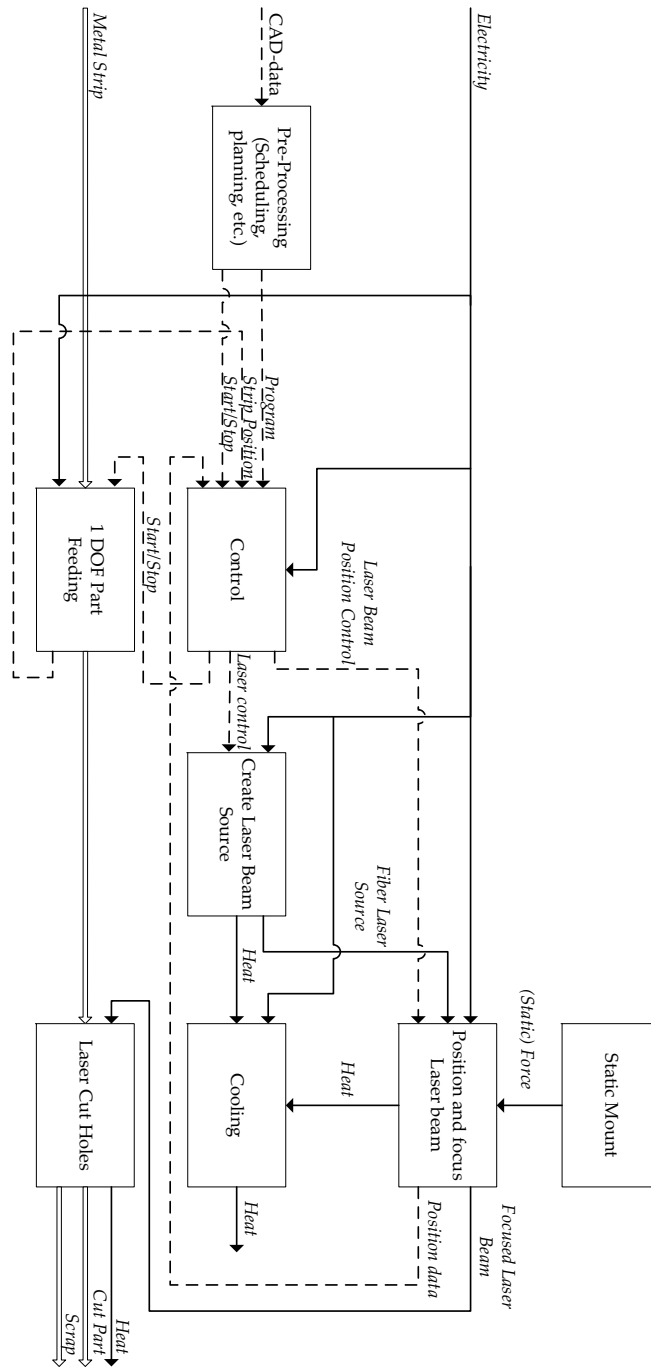


Figure D.1: Functional structure for RLC of holes ahead of or inside the roll forming mill at IAI.

D.2 RLC after the roll forming mill at IAI

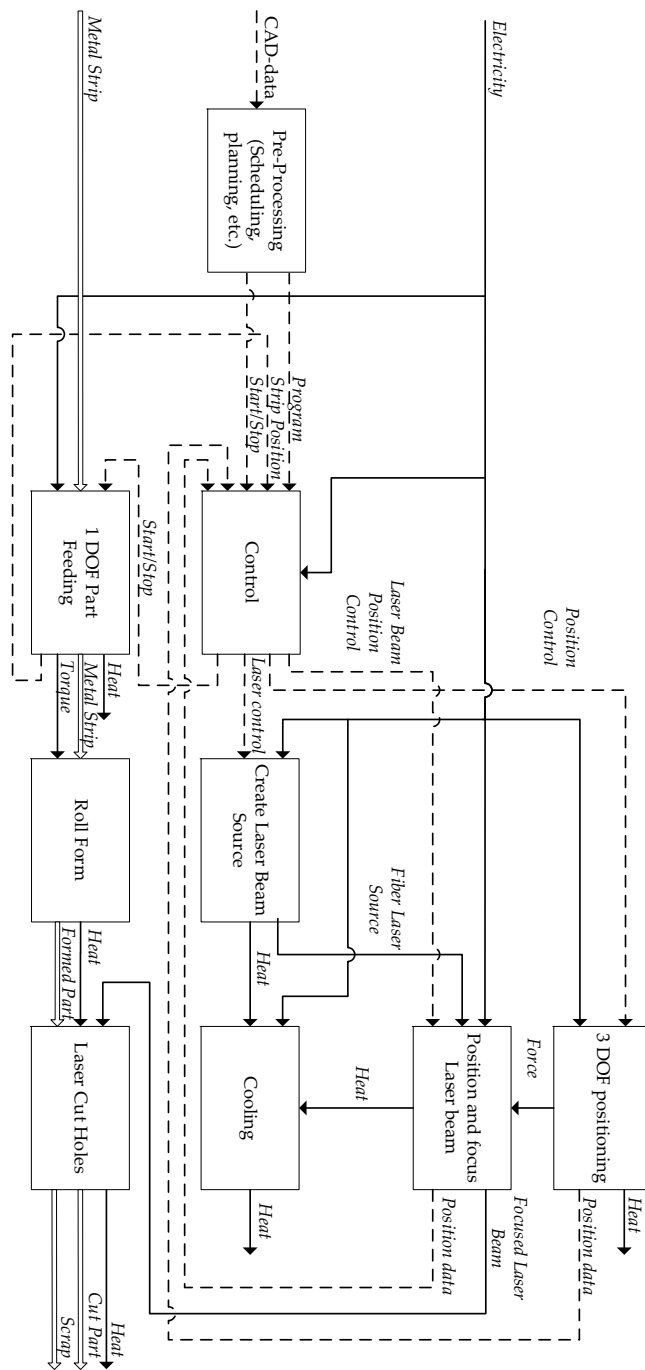


Figure D.2: Functional structure for RLC of holes after the roll forming process at IAI.

D.3 Robotic RLC at Grundfos

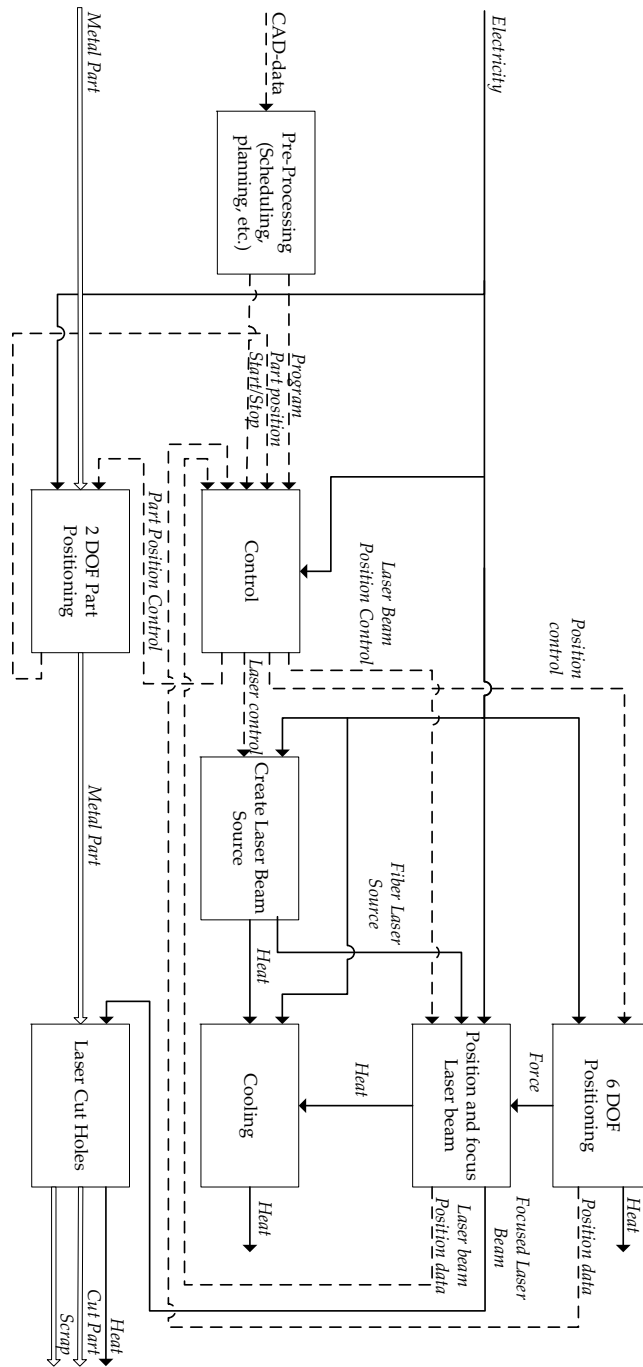


Figure D.3: Functional structure for robotic RLC at Grundfos.

D.4 Station RLC at Grundfos

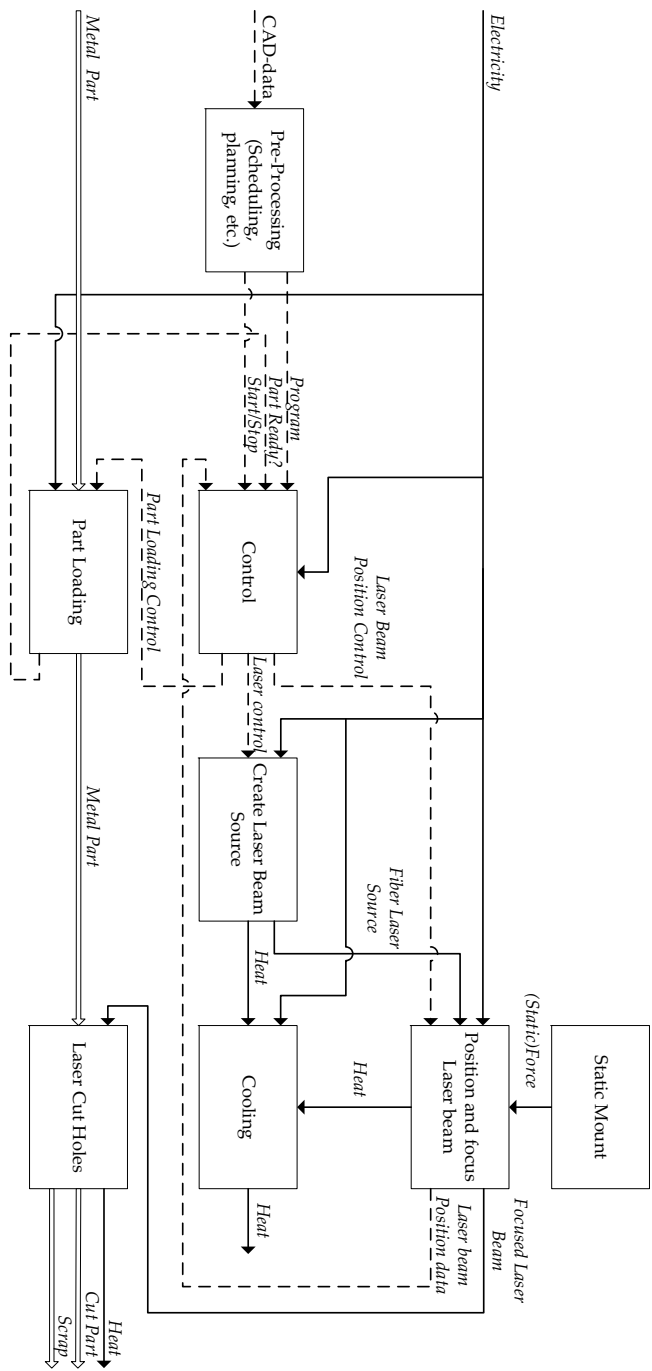


Figure D.4: Functional structure for using RLC as a station in a production line at Grundfos.

D.5 Robotics RLC of holes in car bodies at Volvo

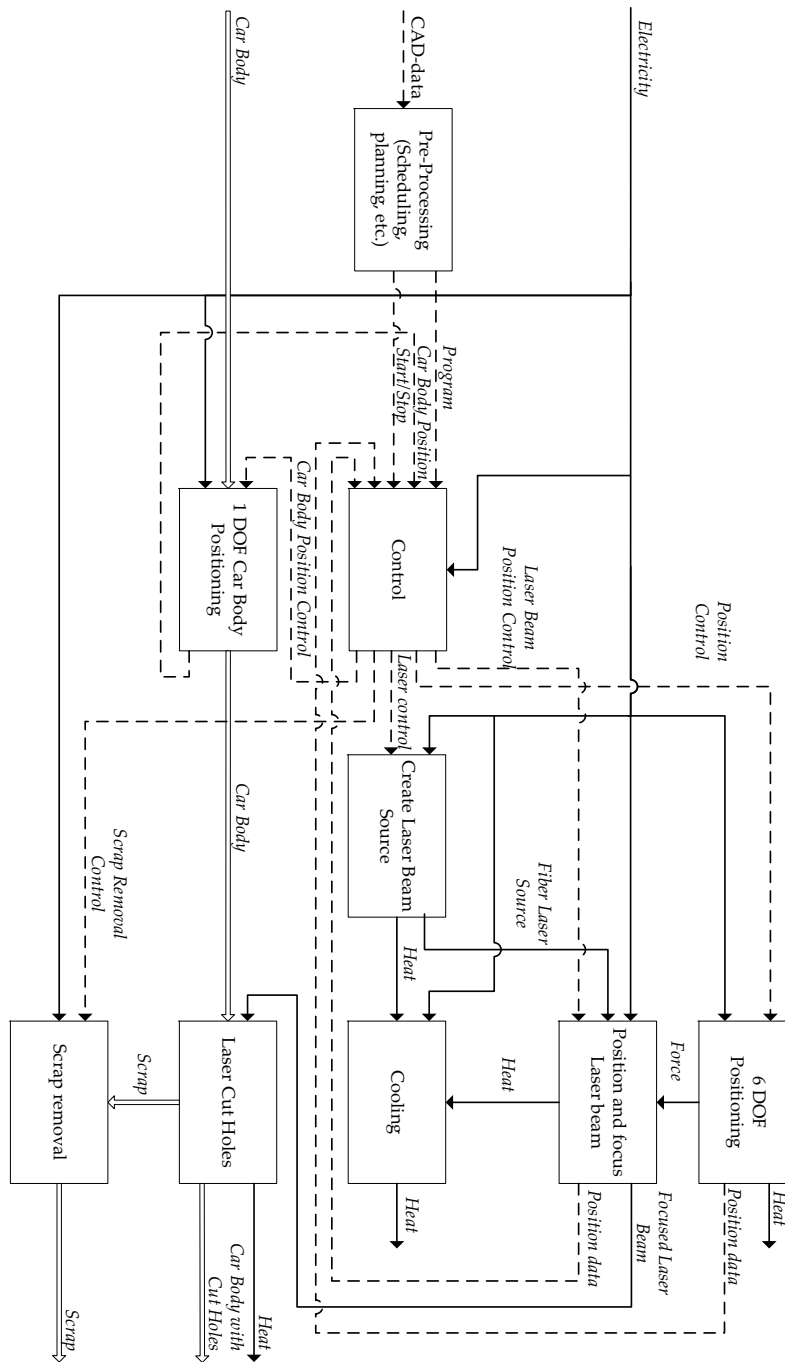


Figure D.5: Functional structure for the cutting of holes in car bodies at Volvo using the RLC process.

APPENDIX E

Requirements

E.1 Functional Requirements

Name	FR-3: Selecting variants in random order.
Summary	The user shall be able to review modules and select variants in a random order. There are in other words no predefined sequence for the user to select variants.
Rationale	When the users are able to reselect all variants at any time the users will quickly be able to test and try out different combinations of variants. In the end this should make it faster to configure a complete system.
Requirements	The system shall take into account that any module can change variant at any time and warn the user if the change clashes with another module.

Table E.1: *Functional Requirement 3.*

Name	FR-4: View and edit user info and pending inquiries.
Summary	The user shall be able to view and edit user info and pending inquiries. The user shall further more be able to delete inquiries if they no longer are of interest to the user. Finally the user can delete his/her account.
Rationale	If a user changes address, phone number or similar it is important that the user can change these information so that the ROBOCUT team always can get in contact with users. It is also important that the user can delete pending inquiries that no longer are of interest so that the ROBOCUT does not use time on inquiries that have no real interest to the customer.
References	A login system is required, see non-functional requirements: security

Table E.2: *Functional Requirement 4.*

Name	FR-5: Notifying user upon successful inquiry with inquiry ID.
Summary	The system shall notify the user upon a successful sent inquiry by sending an email to the user containing at least an unique ID for the specific inquiry.
Rationale	The user may want to consult the ROBOCUT team about an inquiry and the unique ID will help the ROBOCUT team identify the specific inquiry.
Requirements	Each inquiry shall dynamically be given a unique ID.

Table E.3: *Functional Requirement 5.*

Name	FR-6: Showing details of selected variant.
Summary	The system shall show details (image and description) of the currently selected module variant.
Rationale	The details will enable the user to further study a variant before finally deciding. This functionality will also serve to remind the user of what variant is currently selected.
Requirements	If the user selects a variant and then alters the requirements the selected variant may no longer be displayed as a choice to the user. In this case the configurator should still be aware of what variant the user selected and display the details for this variant.

Table E.4: *Functional Requirement 6.*

E.2 Non-Functional Requirements

Name	NFR-3: Warn user immediately with informative text when selecting incompatible variant of module.
Summary	If the user selects a variant of a module that is incompatible with another selected variant of module the user should be warned immediately with informative text.
Rationale	The informative text should allow the user to understand why they are not compatible and revise the selecting based on this information. This should in the end lead to the user being able to understand the relation between the modules better.

Table E.5: *Non-Functional Requirement 3.*

Name	NFR-4: Management of variants.
Summary	It should be possible for the administrator to edit, add and delete variants of modules.
Rationale	This will make it possible to keep all variants up to date using an online management system which is much easier than having to update the whole database offline and then having to upload the entire database.

Table E.6: *Non-Functional Requirement 4.*

Name	NFR-5: Usability (by target user community).
Summary	The configurator should present the user with help and information mostly concerning the user specified requirements.
Rationale	The user may be uncertain of way a specific requirement exactly does and refers to. In such case the user should be able to easily and intuitively get help and information about that exact requirement.

Table E.7: *Non-Functional Requirement 5.*

Name	NFR-6: Security.
Summary	A secure and reliable user account system shall be implemented. The system should be secure enough for users to trust it to hold sensitive user information. User passwords should be encrypted.
Rationale	A secure user account system is a must for users to trust the configurator to hold sensitive information from users.

Table E.8: *Non-Functional Requirement 6.*

Name	NFR-7: Accessibility.
Summary	The system should be made available to as many as possible. This should result in using technologies that are common and does not require special software or hardware.
Rationale	The configurator is meant to be a tool for sharing the possibilities of remote laser cutting. It is thus a key factor that as many users as possible have access to the configurator.

Table E.9: *Non-Functional Requirement 7.*

APPENDIX F

User Manuals

This chapter contains a user manual for the scheduling interface and the ROBOCUT configurator. The manual presents screen shots along with brief descriptions.

F.1 Installing and running the ROBOCUT Add-in

In order to use the add-in for Autodesk Inventor 2011 it needs to be installed and registered. This can be done either automatically through Microsoft Visual Studio 2008, or by manually registering the Add-in .dll. Both methods are described below, and both of these assume that Autodesk Inventor 2011 is already installed.

F.1.1 Installing the add-in through Visual Studio

This is the easiest, and most reliable way of installing the add-in. In addition it also provides the possibility of looking through the source code and continue the development. Of course this entails that Microsoft Visual Studio 2008 is installed.

The add-in can be installed as follows (on a x86 system):

1. Open the `ROBOCUTAddIn.sln` C# solution located in the folder *Part I/Inventor Add-in* on the enclosed CD with Microsoft Visual Studio 2008
2. Once the solution is loaded, go to "Project -> ROBOCUTAddIn Properties" or press Alt+F7, this should open the options window.
3. Go to the "Build" tab, and make sure that the "Active" configuration has been chosen and that the "Register for COM interop" option is selected.
4. Once this is done, the Add-in needs to be built. This is done by choosing "Build -> Build solution" or F7.
5. If the steps above have been carried out correctly, and the build was successful the add-in should be ready for use with Inventor!

F.1.2 Installing the add-in by manually registering the .dll

This method of installing the add-in is more difficult, and it is not guaranteed to function properly in all cases as the manual registration procedure is somewhat system dependent.

The add-in can manually be installed as follows (on a x86 system):

1. Navigate to the folder *Part I/Inventor Add-in* on the enclosed CD.
2. Copy the entire folder to a desired location on the local harddrive (e.g. "C:/ROBOCUTAddIn").
3. Once copied navigate to the folder on the local harddrive and run the file: "x86 Register.bat" in administrator mode.
4. This should register the location of the Add-In, and it should be ready for use in Inventor.

Note that this guide has been tested on Windows 7 only. If the Add-In is not usable after the above procedure, it is advised to visit Autodesk Inc. [2011] for assistance on how to properly register the add-in.

F.2 Using the ROBOCUT Add-in

This section provides a quick walkthrough of the functionality of the ROBOCUT add-in. The walkthrough presents all of the steps necessary to define the cutting tasks and process parameters, run the scheduler and finally open and visualize the results based on the example part seen in Figure A.5.

As a very first step it the add-in must be installed as described in the previous section. Once installed, Autodesk Inventor 2011 is started, and the sheet metal part that needs to be scheduled is opened. Once the part has loaded, the "Flat Pattern" environment is selected next, as shown on Figure F.1:

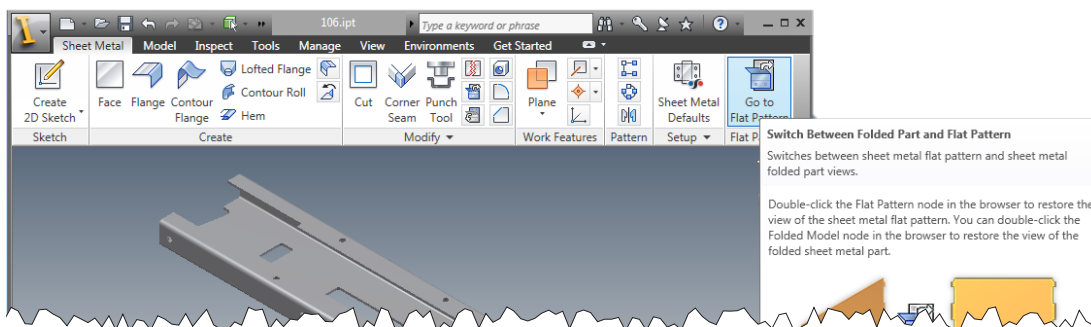


Figure F.1: A sheet metal part is opened, and the Flat Pattern environment is selected.

Once in the "Flat pattern" environment, the ROBOCUT panel should appear along with the "Scheduler" button for starting the add-in as seen in Figure F.2. If this is not the case, the add-in has not been installed correctly.

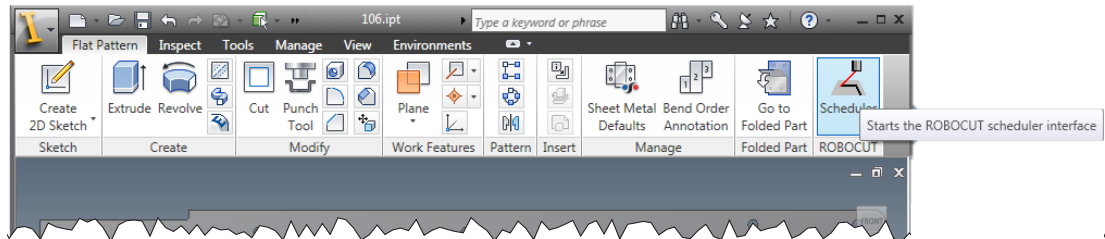


Figure F.2: The ROBOCUT Add-In button should be visible after entering the Flat Pattern environment.

Once the Add-in has been started the ROBOCUTAddIn window should appear as seen in Figure F.3, and it is ready for use.

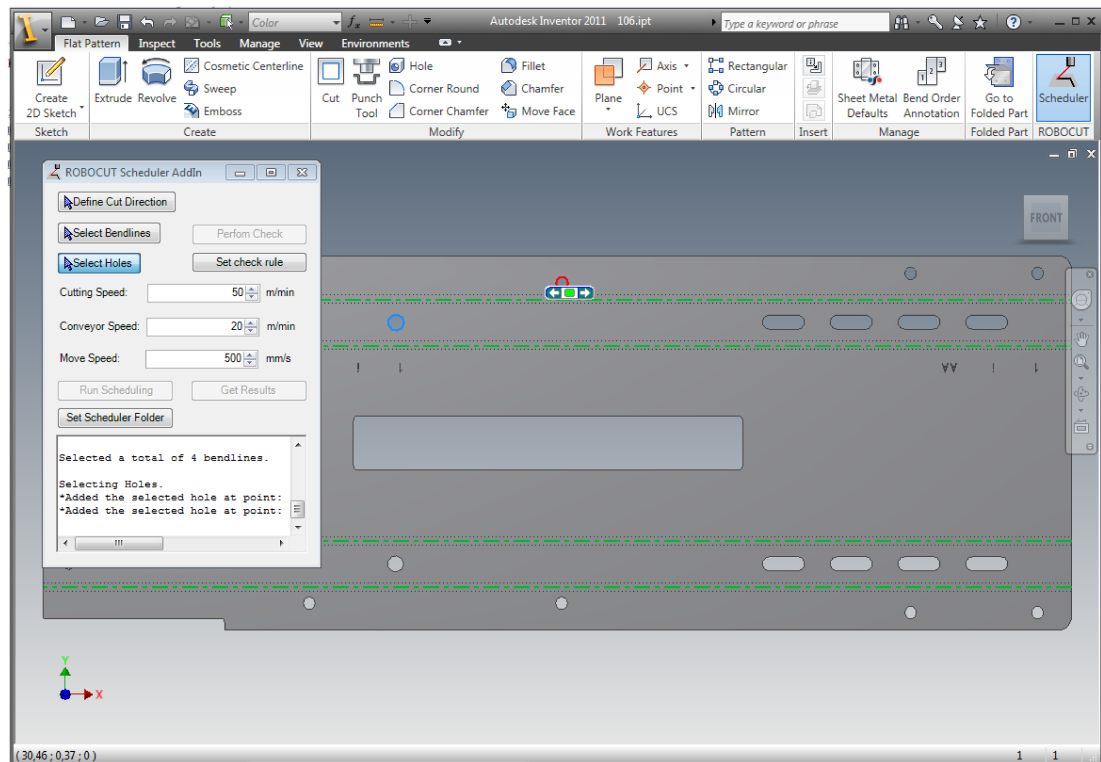


Figure F.3: When the ROBOCUT Add-In it will open a separate window.

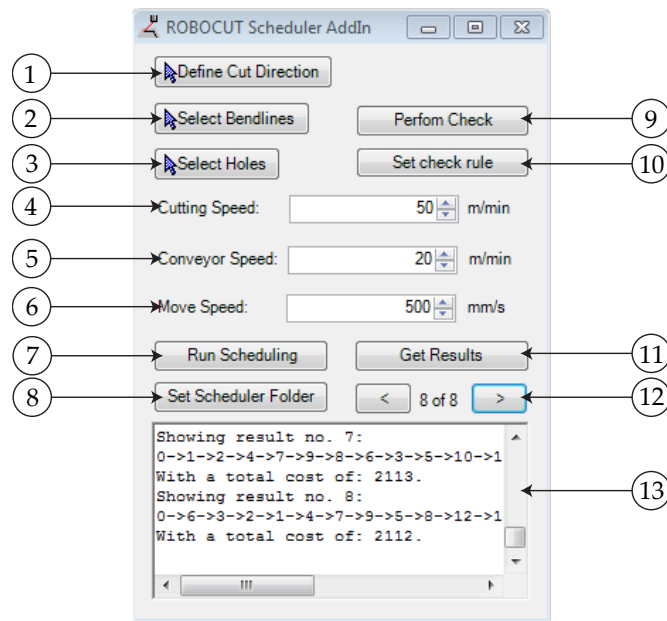


Figure F.4: Close-up of the ROBOCUTAddIn window.

In what follows, the functionality of the Add-In will be described based the numbered Figure F.4.

1. *Define Cut Direction.* This is the first and only button that is active when the Add-In is first opened. It is used to describe the direction of which the conveyor moves the part during the cut. The direction is defined by first choosing the leading edge of the part, and then an edge perpendicular to the leading edge.
2. *Select Bendlines.* When the cut direction has been defined, this button becomes active. In order to perform a check (9) if any holes are to close to a bendline, the bendlines (shown in green in Figure F.3) needs to be selected. When the button is pressed, it is only possible to select the lines parallel to the direction of travel.
3. *Select Holes.* The next button to become active. This allows the user to select the holes, and holes only, one by one, as seen in Figure F.3. Based on the selection the Add-In selects an appropriate starting point for the cutting task, and extracts the length of the cut path.
4. *Cutting Speed.* This box allows the user to specify the parameter for the laser cutting speed.
5. *Conveyor Speed.* This box allows the user to specify the parameter for the conveyor speed, that is the speed at which the metal strip moves in the cut direction specified in (1).
6. *Move Speed.* This box allows the user to specify the parameter for the move speed, that is the linear speed in the cutting plane that the laser beam can travel

between cuts.

7. *Run Scheduling.* Once the cutting tasks have been defined through (1)-(3) and the parameters this button becomes active. When pushed, it will export the defined cutting tasks and parameters to a the text files "coordinates.txt" and "constants.txt", respectively and lead the user to the folder containing the scheduling executable. The user can then manually execute the scheduler before returning to Inventor.
8. *Set Scheduler Folder.* Is simply used to indicate the location of the scheduling executable that is used in (7).
9. *Perform Check.* Is an experimental function for checking if any of the selected holes are to close to the selected bendlines. This function is used when all of the holes have been selected, prior to running the scheduler.
10. *Set check rule.* Sets and edits the rule used in (9).
11. *Get Results.* Once "Run Scheduling" has been executed, this button becomes active. When pressed it looks for the file "result.txt" and loads the results it contains. The loaded results are then loaded, and the last solution path is drawn onto the part, as seen in Figure F.5.
12. *Scroll through results.* In case multiple solutions have been found, the user is able to scroll through them using these navigation buttons.
13. *Console window.* During the use of the add-in, the console window provides some detailed information about the commands that are carried out, as well as the results of the calculations carried out by the add-in. Once the add-in is closed, this information is saved in a log file.

F.2 Using the ROBOCUT Add-in

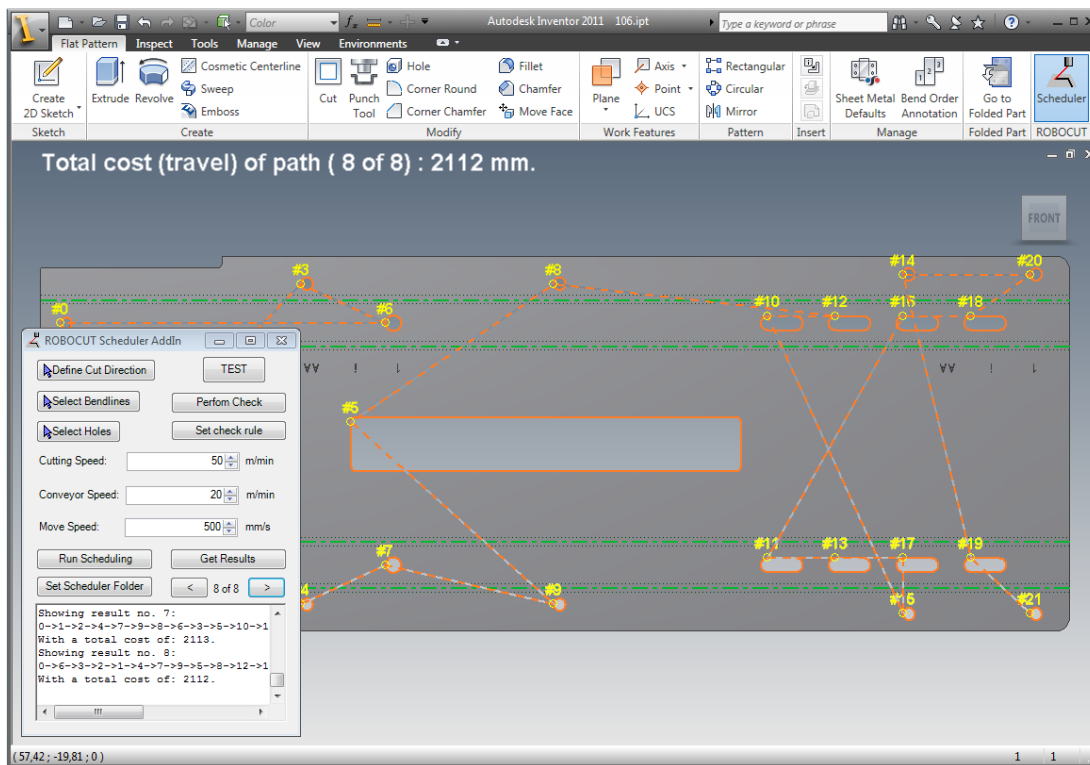


Figure F.5: After running the scheduler, the result can be loaded by pressing "Get Results".

F.3 User Manual for ROBOCUT Configurator

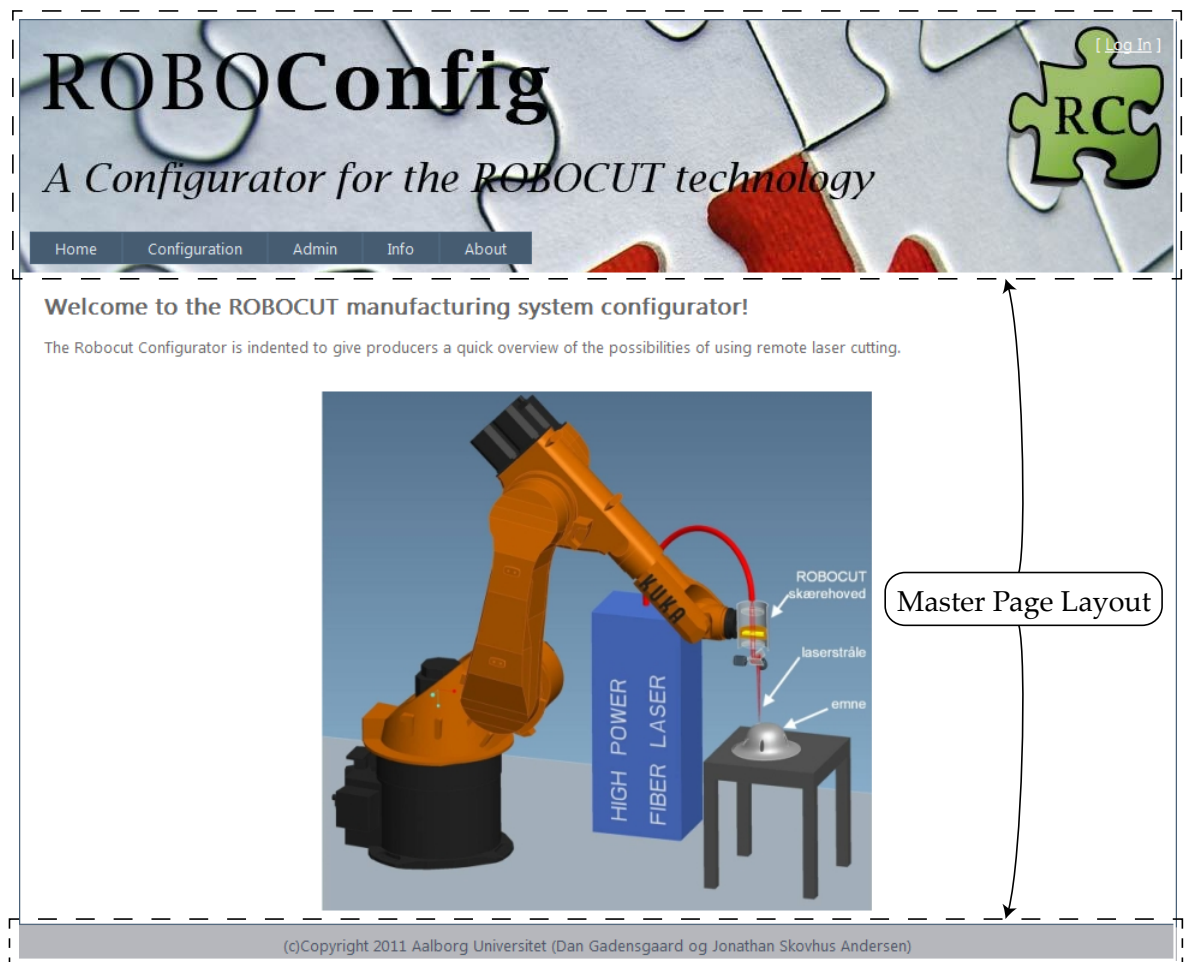


Figure F.6: Screen shot of "Home.aspx".

The default page display a welcome message and a picture of the ROBOCUT concept. The screen shot of the default page shows the master page layout. The master page layout is visible for all pages and will not be shown in the following screen shots.

Log In

Please enter your username and password. [Register](#) if you don't have an account.

Account Information

Username:

Password:

☐ Keep me logged in

Log In

Figure F.7: Screen shot of "Login.aspx".

If not logged in, the user will be redirected to the login page when trying to access "Configuration", "Admin" or "Info". The login page redirects the user to the page the user tried to access when successfully logged in. The login page contains a link that leads to the register page.

Register

Registration Information

User Information 1/2:

First Name*:

Last Name*:

Address 1*:

Address 2:

City*:

Postcode*:

Region/State*:

Country*:

Phone number*:

Position*:

Company*:

CVR:

First name is required x

Next

Figure F.8: Screen shot of "Register.aspx", step 1.

Register

User Information 2/2:

Username:

Password:

Confirm Password:

Email:

Previous **Create User**

Figure F.9: Screen shot of "Register.aspx", step 2.

Registration consists of two steps. Step one collects additional user information. Step two collects user name, password and email address.

Fiber Laser

Movement of Scanner Head

Movement of Part

Control

Movement of Scanner Head

User Specified Requirements

Tolerance

0.35 mm

Cut/Weld Geometry

☐ Planar
 ☐ 3D Planar
 ☒ 3D Complex

Size of Geometry to Cut/Weld [mm]

Height: 600

Width: 500

Length: 100

Evaluate

A robot or linear robot with 4 to 6 axes are recommended for this setting.

☐ Static Mount

Movement of Part or Robot

Gantry not needed.

Select Gantry

☐ ABB IRBT 4004
 ☐ KUKA KL 1000-2
 ☒ None

Part movement

☐ Gantry
 ☐ Robot

Robot

Sort by speed

Sort by reach

☒ ABB IRB 4600-45
 ☐ KUKA KR 60HA
 ☐ ABB IRB 4600-60
 ☐ FANUC M-710iC50
 ☐ FANUC M-710iC50S
 ☐ ABB IRB 4400-60
 ☐ KUKA KR 60-3
 ☐ KUKA KR 60-3 L45

Movement	XYZYPR
Repeatability	0,05
Speed	1385
Mountingpositions	Floor, shelf, inverted or tilted
Weight	412
Numberofaxes	6
Price	0

Configurator selects groups of variants based on requirements

Linear Module


Sort by speed

Sort by stroke

☐ Güdel FP5
 ☐ Schunk PLB110 90 70 PR70

Send Inquiry...

Details for ABB IRB 4600-45



Show Details... Show/Hide

Selected Setup

Fiber Laser: None
Scanner Head: None
Scanner Movement: **ABB IRB 4600-45**
Gantry: **None**
Part Movement: **None**
Part Loading: **None**
PC Controller: **None**
Robot Controller: **None**

Figure F.10: Screen shot of "Configurator.aspx".

The configurator page contains a tab for each module. A screen shot of the configurator page with the "Fiber Laser" tab open is shown in Section 13.1, Figure 13.1. The Figure here shows the content of the "Movement of Scanner Head" tab. The green text shows the user what groups of variants are recommended for the input in "Cut/Weld Geometry". Only the recommended variants are displayed in the mid column.

F-10

F User Manuals

Add/Edit Modules

- Title: IPG YLS 1000CL
 Power: 1000
 Wavelength:
 image: YLS-1000-CL.jpg
 Thickness:
 Description: Each classification of high power fiber laser serves a targeted application demographic. With a larger feed fiber core and simple design, the cladding laser (CL) is aptly named. Also utilized for micro and macro welding, brazing, annealing and heat treating applications, the YLS-CL is considered a direct competitor to direct diode systems. With its maintenance-free, robust package the YLS-CL allows for plug and play fiber delivery with interchangeable process fibers available in either square or round diameters up to 1 mm.
- Title: IPG YLS 1500
 Power: 1500
 Wavelength:
 image: YLS_Series.jpg
 Thickness:
 Description: Each classification of high power fiber laser serves a targeted application demographic. The most versatile and customizable option within the product line is the YLS series fiber laser. Developed as a complete system, this design features the widest range of fiber diameters, the option to terminate to up to 6 ports from one power source, and the ability to upgrade
- Title: ROFIN FL030
 Power: 3000
 Wavelength:
 image: image-not-available.jpg
 Thickness:
 Description:
- Title:
 Power:
 Wavelength:
 image:
 Thickness:
 Description:

Figure F.11: Screen shot of "Admin.aspx".

The administrator section of the web site makes it possible to add/edit variants. Each module has a tab. The variants are displayed in a bulleted list each with an edit and delete button. In the bottom there is text fields for adding a new variant.

User Accounts

	Email	Username	Last Login Date	Create Date
Delete	ksh@ipu.dk	ksh-at-ipu-dot-dk	03-06-2011 17:16:37	03-06-2011 17:16:37
Delete	ksh@ipu.dk	ksh	03-06-2011 17:16:19	03-06-2011 17:16:19
Delete	ksh@ipu.dk	ksh@ipu.dk	03-06-2011 17:16:53	03-06-2011 17:15:58
Delete	jonathanskovhus@gmail.com	admin	06-06-2011 13:20:12	01-06-2011 15:06:48
Delete	foo@ipu.dk	flol	01-06-2011 10:33:37	01-06-2011 10:33:37
Delete	morten@m-tech.aau.dk	morten	30-05-2011 08:40:01	30-05-2011 08:40:01
Delete	jonathanskovhus@gmail.com	jonathanskovhus	15-05-2011 21:38:21	15-05-2011 21:38:20

UserId

e3808688-a1fc-4bec-9960-9e07122dec2c

UserName

admin

FirstName

admin

LastName

test

Address1

test

Address2

City

test

Region

test

Postcode

9000

Country

test

Phone

12345678

Position

test

Company

test

CVR

12345678

Edit

Username:

admin

Get User

Hide User

Pending Inquiries

SystemID	b1e7958e-bc05-440b-8d7d-c1a6757860f1
UserId	6e8e8b1c-95f6-40a3-bceb-8fa0d7c0a4d6
Username	ksh@ipu.dk
FiberLaser	ROFIN FL040
ScannerHeadMovement	KUKA KR 60HA
ScannerHeadType	Cutting
Gantry	None
PartMovement	Rotary Table
PartLoading	Automatic
PCController	PC Controller
RobotController	KUKA KRC2
SubmitDate	03-06-2011 07:06:13
Edit Delete	

1

2

3

Figure F.12: Screen shot of "Admin.aspx".

The administrator section of the web site also contains a "Users" tab. This tab display a list of all user accounts and make it possible to get the additional information about a specific user. In the button all inquiries are shown.

Here's your info, admin :

Username	admin
First Name	admin
Last Name	test
Address 1	test
Address 2	
City	test
Region	test
Country	test
Postcode	9000
Phone	12345678
Position	test
Company	test
CVR	12345678
Edit	

Pending Inquiries

SystemID	3eb4c1f3-035f-404f-bf15-39378625b710
FiberLaser	ROFIN FL040
ScannerHeadMovement	FANUC M-710iC50
ScannerHeadType	Cutting
Gantry	None
PartMovement	None
PartLoading	None
PCController	None
RobotController	None
SubmitDate	02-06-2011 12:06:34
Delete	

Figure F.13: Screen shot of "Settings.aspx".

The info page displays the user account information of the user currently logged in. In the bottom of the page, the user can see all submitted inquiries.

About

This product configurator has been developed as a part of a Master's Thesis at Aalborg University in Spring 2010. The website is developed using ASP.NET 3.5 C# in Microsoft Visual Studio.

- Dan Gadensgaard
- Jonathan Skovhus Andersen

Only visible if
the user is
logged in

Your feedback:

Test...

Thank you for submitting feedback!
Your feedback will be reviewed as soon as possible.

Figure F.14: Screen shot of "About.aspx".

The about page can be accessed without being logged in. However the send feedback function will not be visible if the user is not logged in.