A methodological approach for optimisation of product development processes by application of design automation

Master of Science Thesis

A. Mulder BSc

Delft University of Technology

TU Delft

A METHODOLOGICAL APPROACH FOR OPTIMISATION OF PRODUCT DEVELOPMENT PROCESSES BY APPLICATION OF DESIGN AUTOMATION

MASTER OF SCIENCE THESIS

by

A. Mulder BSc

in partial fulfillment of the requirements for the degree of

Master of Science
in Aerospace Engineering

at the Delft University of Technology

Thesis registration number: 037#15#MT#FPP

TUDelft Delft University of Technology

KE-works

DELFT UNIVERSITY OF TECHNOLOGY
FACULTY OF
AEROSPACE ENGINEERING
DEPARTMENT OF
FLIGHT PERFORMANCE AND PROPULSION

The undersigned hereby certify that they have read and recommend to the Faculty of Aerospace Engineering for acceptance a thesis entitled **"A methodological approach for the optimisation of product development processes by application of design automation"** by **A. Mulder B.Sc.** in partial fulfillment of the requirements for the degree of **Master of Science**.

Dated: 14th of August, 2015

Head of department: ______________________

Prof. dr. ir. L. Veldhuis

University Supervisor: ______________________

Dr. ir. G. La Rocca

Industry Supervisor: ______________________

Dr. ir. E.J. Schut

Reader: ______________________

Dr. ir. W. Verhagen

ACKNOWLEDGEMENTS

This thesis is the final deliverable concluding my time as a graduate student. The past seven years I have learned a lot of things, but most importantly I enjoyed it. From the first year lectures to the Design Synthesis Exercise and from the internship assignments to writing this thesis. Besides learning a lot about aerospace engineering, I learned a lot of other things. I think this is the power of the Delft University of Technology, to provide high quality education in combination with the academical freedom to explore other areas inside and outside of the curriculum.

Although solely my name is written on the cover of this thesis it does not mean I did everything by myself. Quite the opposite actually, I could not have achieved this goal without the help of many others. I want to express my gratitude to all of you who supported me in the past seven years. I also would like to thank some people in particular due to their substantial contribution to either my academical work, my mental state or both.

First of all, I would like to thank my supervisors *Gianfranco La Rocca* and *Joost Schut*. Although the both of you have packed schedules you always managed to free up time for discussions and feedback. I think it is a good sign that most of the meetings ran overtime. They were both useful and cheerful. Gianfranco, thank you for your sharp analysis and detailed feedback on deliverables and especially the conference paper. Joost, many thanks for guiding me on a daily basis and your seemingly unlimited optimism.

Thanks to *Wim Verhagen* for multiple discussions from the very start of this research until the end. I am thankful for these discussions and your review of the conference paper. I am glad you were willing to join the thesis committee to assess my work. Thanks to *Leo Veldhuis* for being the chairman of the committee and assessing my work.

Most of the working hours were spent at the KE-works office in a pleasant working environment. Thanks to all the *colleagues* for all their help. The discussions on product development were helpful but also the great supply chain management of coffee and relaxation with fußbal or table tennis increased the quality of this thesis. I would like to thank *Wilco Schoneveld* in particular for his help in getting familiar with the Python language and SimPy package.

I owe many thanks to various people for reviewing my work over the past seven months. Thank you *Rob, Max, Jaap, Wieger and Kevin.*

And last but certainly not the least I would like to thank my *friends and family*. Whenever I was down you always cheered me up and when I was happy you shared this joy with me. I know this sound cheesy, but the process of writing this thesis was one with many personal highs but unfortunately some lows. It is great to see what friends and family mean at these moments. I want to thank *Ruth* in special, you have been, without a doubt, the best motivator in the past years and will be for the coming year. Thank you!

A. Mulder BSc
Delft, August 2015

SUMMARY

In the past decades a clear transition can be seen from fully human-based production techniques towards more automated systems. With the Product Development Process (PDP) being a potential source of competitive advantage, the same trend of adopting more automation can be seen in the PDP. The application of automation can lead to large reductions in process lead time. The reduced lead time can be used to enhance product performance by assessing more design options in the same time, or to reduce the cost-of-delay and obtain a larger market share. It is clear that a reduction in lead time is worth an investment for companies and therefore automation can be adopted to improve the PDP performance.

Both industry and academia acknowledge the lack of a quantitative method to assess the effect of the application of automation on the performance of a given process. Furthermore no methods are available to assess the effect of incremental automation. Another limitation is that most models investigating automation in the PDP, only take into account a binary type of automation: either fully human or fully automated.

This research aims at addressing these gaps of knowledge and the goal of this research is to develop a tool that provides more insight in the costs and benefits of automation in the PDP taking into account incremental automation and various levels of automation.

To achieve the objectives a novel methodology is developed i) to model any PDP as a combination of a predefined set of specific activities and ii) to define different levels of automation for these activities. Subsequently, metrics are developed to measure the impact of different levels of automation on different activities, both in terms of activity lead time reduction and implementation cost. A simulator using Discrete Event Simulation (DES) is developed which utilises the proposed process model and metrics to analyse, among others, the lead time, automation investment cost and process cost for the overall process (for a given process architecture). Finally, the simulator is connected to an optimiser which tries to find the most convenient level of automation for each of the PDP activities, in order to generate the Pareto front qualitatively illustrated in Figure 1. Here each Pareto optimal solution corresponds with a process architecture where each activity has a specified level of automation as illustrated in Figure 2.

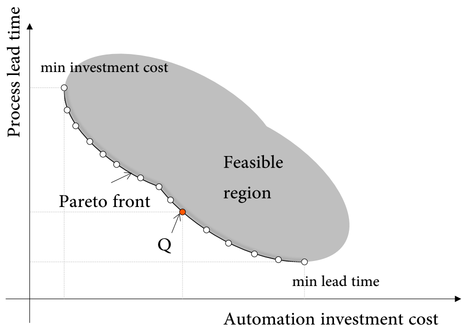


Figure 1: Qualitative representation of a Pareto front trading off lead time and investment cost

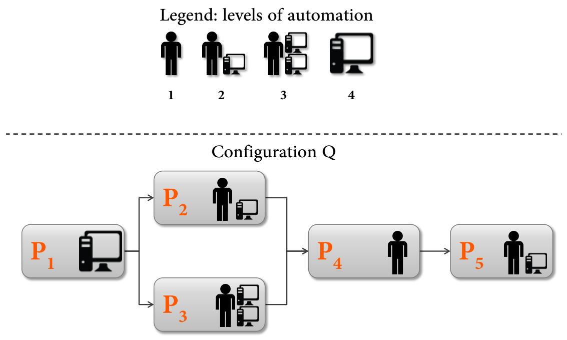


Figure 2: Simplified illustration of the process architecture Q on the Pareto front

The proposed methodology is extensively verified. Verification is performed for both the simulator and the optimiser. Based on this verification it can be concluded that with appropriate settings the optimiser manages to find Pareto optimal solutions for different levels of automation. Due to scarce validation data, the methodology has been validated based on expert judgement.

The proposed methodology and the analysis and optimisation framework are demonstrated by application to an industrial case study. The case study concerns the conceptual design phase of an aircraft component, performed by a multinational aerospace enterprise. The case study successfully demonstrates the feasibility and applicability of this methodology and accompanying frameworks. Multiple Multi-Objective Optimisations (MOOs) are performed to trade off various objective functions. In this case study, specific activities in the process are identified to be more effective to automate. Results show that, compared to the status quo, an investment in the lug sizing tasks of 7,1% of the investment cost of full process automation can lead to a potential lead time reduction of more than 40%. The proposed methodology proves to be successful in objectively quantifying the costs and benefits of automation in the PDP and subsequently selecting the optimal automation level.

Contents

List of Figures

List of Tables

List of acronyms

AI	Artificial Intelligence.
BEP	Break Even Point.
CAD	Computer Aided Design.
CATIA	Computer Aided Three-dimensional Interactive Application.
CM	Cost Matrix.
DA	Design Automation.
DES	Discrete Event Simulation.
DM	Duration Matrix.
DSM	Design Structure Matrix.
GA	Genetic Algorithm.
IQ	Information Quality.
KBE	Knowledge Based Engineering.
KPI	Key Performance Indicator.
LC	Learning Curve.
MDO	Multi-Disciplinary Optimisation.
MOO	Multi-Objective Optimisation.
NSGA-II	Non-dominating Sorting Genetic Algorithm.
PDP	Product Development Process.
ROI	Return on Investment.
UML	Unified Modeling Language.
VBA	Visual Basic for Applications.
VT	Vertical Tailplane.
WMS	Workflow Management Software.

1 INTRODUCTION

In the past decades a clear transition can be seen from fully human-based production techniques towards more automated systems. This transition is focused on reducing lead time, decreasing process cost and improving product consistency and quality. The same trend of adopting more automation can also be seen in the Product Development Process (PDP), driven by a growing focus on PDP improvement as a potential source of competitive advantage [6]. In particular, for many companies lead time duration is the most important performance measure of the development process, because a reduced time-to-market (i.e. reduced lead time) results in a reduction in cost-of-delay and a larger market share [7]. This reduction in lead time can also be used to increase product quality by investigating multiple design options in the same lead time as without automation. Because of these benefits a reduction in lead time is worth an investment for companies. Design Automation (DA), Knowledge Based Engineering (KBE), Artificial Intelligence (AI) and Computer Aided Design (CAD) are examples of computer based technologies adopted to improve the PDP.
This chapter will first provide a brief discussion on the need for automation in product development in Section 1.1. Subsequently the current challenges faced in the field of process optimisation by means of automation are discussed in Section 1.2. Subsequently in Section 1.3 the research objectives, questions, approach and scope are discussed. Section 1.4 provides a quick summary of the involved research partners. Finally in Section 1.5 the content of the remainder of this thesis is outlined.

1.1. THE NEED FOR AUTOMATION IN PRODUCT DEVELOPMENT

The way the PDP is executed in industry is constantly evolving. Over the years, a paradigm shift can be observed, where starting from a purely technical process, the PDP is now integrating both technical and business process aspects. The overall process or product performance is becoming increasingly important. Hence the total performance should be taken into account during the PDP and thus more than just the technical disciplines need to be taken into account. The total life-cycle cost of a product needs to be taken into account as early as possible. Moreover the need for an optimal design becomes increasingly important, especially in industries such as the aerospace industry where minor adjustments in the design can lead to major reductions in total life-cycle cost.
Traditional development techniques lead to a so-called "knowledge paradox" where the designer reduces design freedom by gaining knowledge on the design [8]. This paradox is also illustrated in Figure 1.1. This paradox has resulted in product development techniques like concurrent engineering and Multidisciplinary Design Optimisation. Processes have evolved from stage-gated sequential engineering models to the iterative concurrent models.
These new process models have proven to reduce lead time and often increase overall process performance [9]. On the other hand these processes have an increased risk for rework and the amount of iterative loops increases significantly according to Clark and Fujimoto (as cited by Terwiesch *et al.* [9, p. 404]). This effect of iteration and rework leads to more repetitive tasks in the PDP. Automation initiatives, like DA and KBE, have proven to be valuable for repetitive, non-creative tasks [8]. Computers excel in the execution of repetitive work. Automation reduces the time spent by a human resource on these repetitive tasks and often results in shorter lead times and increased productivity [11].
By adopting these processes with an increase in the amount of iterations and rework automation is essential.

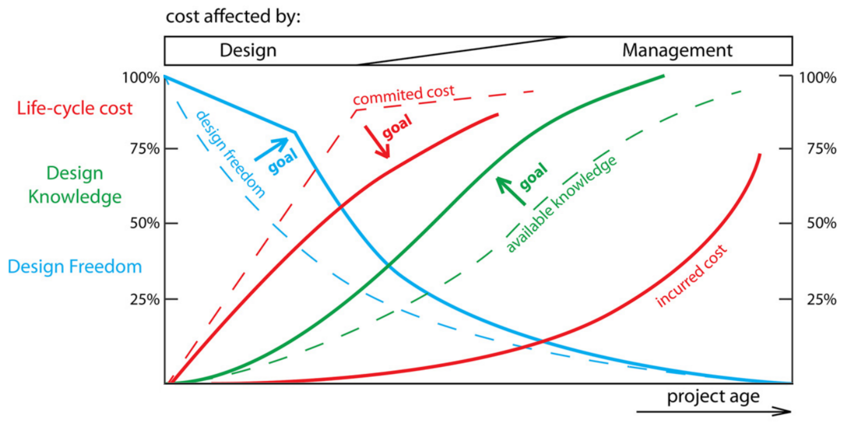


Figure 1.1: Qualitative representation of the "knowledge paradox". Solid lines represent the case of traditional development techniques and the dashed lines represent the need to bring knowledge forward in the process and increase the span of design freedom. [1]

Without automation either too few iterations can be made to deliver an optimised product, or the lead time increases significantly resulting in a high cost-of-delay and reduced market share [7].
The value of automation does not solely rest in the reduction in lead time due to the computational power of a computer. Automation also leads to consistent quality; the outcome is based on specific algorithms and consistent use of the same knowledge. Furthermore, by the application of automation the cost of the human resources spending time on an activity are reduced or even eliminated. More background information on the benefits of automation can be found in Section 2.5.4.

1.2. CHALLENGES IN THE APPLICATION OF AUTOMATION

Automation is in literature often regarded as a binary option for process improvement, meaning that a process either is fully automated or not at all. This is not a realistic point of view since automation can be seen as an incremental innovation. In practice it is often not possible, or even desirable to automate a full process at once due to technology challenges, but also in due to the human side adoption of the automated solution [12]. Another practical aspect playing in favour of incremental innovation is the available budget of the company. Often a proof of concept is generated before a whole process can be automated. This proof of concept is for example a sub-process which is automated, or the process as a whole which is automated to a lower level of automation than in the full-scale automation initiative. To add to this, the option of different levels of automation is often not assessed. An activity is either performed by a human or by a computer but the option of a human with assistance of the computer is in many cases not assessed.
Furthermore, as discussed by Verhagen *et al.* [13], the cost of automation is not assessed a prio: "There is no formal method to generate a priori cost estimates of the automation process". It is furthermore acknowledged that such a method is required to be able to construct objective cost and benefit analysis for automation opportunities.
Finally it is difficult and often impossible, to predict the impact of single changes in the configuration of a process (e.g. by the introduction of automation solutions for specific tasks) of the overall PDP [14].

Before a company can commit to the development or acquisition of automation solutions, management needs critical information, such as the set of PDP activities to automate first, the expected gain in lead time reduction, the cost associated to the implementation of different levels of automation or to the reconfiguration of the whole process to a specific level of automation. In other words, the company needs to gain insight in the costs and benefits of the application of automation initiatives. Figure 1.2 qualitatively displays the current situation where a company incrementally applies automation without knowledge of the shape of the Pareto front (i.e. with the optimal lead time-investment cost combinations). Only a perceived Pareto front (i.e. bold guestimates based on intuition) with a high uncertainty is available. It can also be seen that often the space of feasible solutions is not known since only specific automation initiatives are investigated; it is impossible to investigate all automation initiatives on a low process modelling level (i.e. on a task or activity level).
Figure 1.3 illustrates the desired situation in which a feasible region is known, consisting of many differ-

ent PDP architectures, each one consisting of the complete sequence of process activities, with their level of automation and employed resources. Within this feasible region, knowledge is available on the optimal solutions on the Pareto.

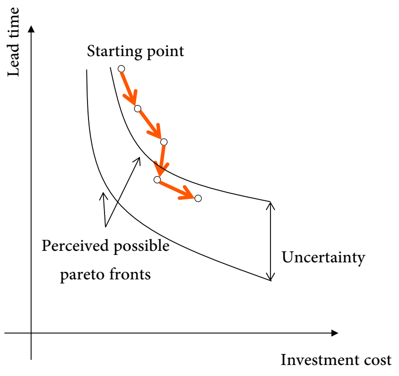


Figure 1.2: Current trade-off between investment cost and lead time in a Product Development Process

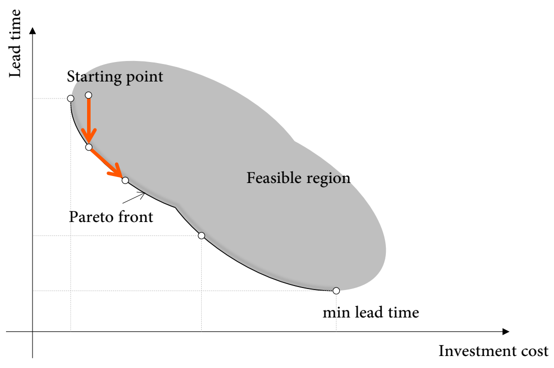


Figure 1.3: Improved trade-off between investment cost and lead time in a Product Development Process

For companies the process architectures on the Pareto front are those of highest interest since they represent optimum combinations of lead time and required investment cost, i.e. PDP architectures for which one of the two objectives (lead time and investment cost) cannot be improved without deteriorating the other. The Pareto front can be used to estimate the investment costs necessary to achieve a certain lead time reduction, or, vice versa, the amount of lead time reduction that can be achieved with a given budget to invest in automation solutions.

1.3. RESEARCH OBJECTIVE, APPROACH AND SCOPE

This section introduces the research objective and sub-objectives, the approach used to achieve these objectives and the scope for this research.

1.3.1. RESEARCH OBJECTIVE

The goal of this research is to create a tool that provides more insight in the costs and benefits of automation in the PDP. This is of great relevance since it addresses the challenges as described in the previous section. The main objective of this research is stated as follows:

Define a methodology to predict the effects of the implementation of automation solutions on the PDP performance by using simulation

This objective can be defined more specifically; the proposed methodology considers the complete process in the current state, hence it does not aims at restructuring it. It evaluates the influence of the application of specific automation initiatives, at single (sub) activity level, on the overall process lead time and investment cost. This leads to the following sub-objectives supporting the main objective:

a Create a framework to model the process on an activity level with varying levels of automation

b Model the impact of different levels of automation on investment cost and activity lead time

c Develop a model to analyse the performance of specified process architectures by the use of simulation

d Develop a method to optimise the levels of automation on an activity level for multiple objectives

e Verify the methodology by means of application on a representative PDP case study

These objectives are subject to the requirement that the developed methodology should operable in a commercial environment.

1.3.2. RESEARCH APPROACH

To achieve the objectives a new methodology is developed (i) to model any PDP as a combination of a predefined set of specific activities and (ii) to define different levels of automation for these activities. Subsequently, metrics are developed to measure the impact of different levels of automation on different activities, both in terms of activity lead time reduction and implementation cost. A discrete event simulator is developed which utilises the proposed process model and metrics to analyse, among others, the lead time and automation cost for the overall process (for a given process architecture). Finally, the simulator is connected to an optimiser which tries to find the most convenient level of automation for each of the PDP activities, in order to generate the Pareto front qualitatively illustrated in Figure 1.3. The proposed methodology and the analysis and optimisation framework are demonstrated by application to an industrial case study. The case study concerns the conceptual design phase of an aircraft component (in particular the study of the rudder-fin connection) performed by a multinational aerospace enterprise. The metrics used in this study to estimate the cost of automation (for various levels of automation) and associated lead time reduction for different types of PDP activities are based on the experience gained by KE-works, in various aerospace PDPs. For the Discrete Event Simulation (DES) and optimisation, SimPy and the Optimus platform are used respectively.

1.3.3. RESEARCH SCOPE

This research is conducted in a limited period of 30 weeks. Therefore scoping of the problem is required, since not every aspect of the research can be tackled to the fullest. The research therefore focusses on the PDPs of complex aerospace products. However this does not necessarily mean that the proposed methodology is not applicable to other industries. The research focusses on a single-company and single-project with multiple departments, although the proposed methodology can be extended as necessary. Furthermore the trade-off as stated in the objective is limited to a trade-off between cost and time. The specific performance indicators are discussed in Section 2.2.

Furthermore, the process of optimisation of the PDP involves many different steps. Based on research by Schut *et al.* [15] and Verhagen *et al.* [13] different steps, as illustrated in Figure 1.4, for process improvement are identified (discussed in more detail in Appendix B).

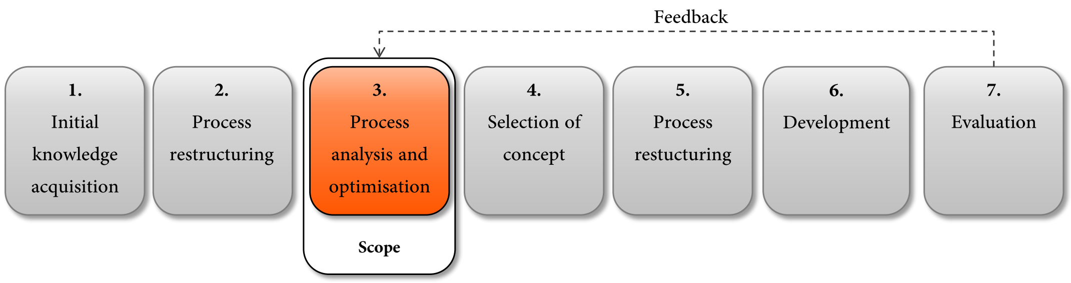


Figure 1.4: Steps involved in process improvement

The primary focus of this research is the step of process analysis and optimisation (step 3). Since knowledge acquisition is required to be able to model the process and subsequently analyse and optimise it, the step of initial knowledge acquisition (step 1) is also part of the focus of this research.
Step 2, process restructuring, is not taken into account in this research. Hence this research investigates the optimisation of a given fixed process without adjusting the sequence of activities (restructuring). Furthermore the steps 4, 5, 6 and 7 are deemed important for the optimisation of the PDP but due to the long lead time of concept selection and subsequent development these steps were not investigated.
Optimisation of a given fixed process without restructuring proves to be of great relevance. Especially in industries with certified processes it is costly, or even impossible, to modify the structure of their PDP, hence the application of automation should be evaluated without restructuring the PDP. However, the authors are aware of the fact that a global and comprehensive optimisation of the PDP cannot be achieved without considering the synergetic effect of automation solutions deployment and process structure restructuring.

1.3.4. RESEARCH ASSUMPTIONS
In this research some assumptions are made. Firstly it is assumed that all processes under consideration in this methodology are recurring and defined processes. Applying this framework is only possible if the process has been repeated before. Defining a process of a constantly changing process is complex, if not impossible. Secondly it is assumed that for the highest level of automation a Workflow Management Software (WMS) is required to integrate the different tasks and activities, replacing the human previously responsible for this integration. Furthermore it is assumed that the level of automation of activities can only increase or remain constant (i.e. no reduced level of automation is possible).
It is assumed that a change in an activity always leads to rework for the subsequent activity, hence the rework probability is equal to 1 for all cases.

1.4. RESEARCH PARTNERS
This research is executed with support of multiple partners in the IDEaliSM project. The IDEaliSM project aims to drastically improve the time-to-market and development cost of high-tech systems and structures [16]. Brief profiles of the involved companies are provided in following paragraphs.

1.4.1. DELFT UNIVERSITY OF TECHNOLOGY
The Delft University of Technology is an internationally renowned institute for high quality research and education. The Flight, Propulsion and Performance department of the Faculty of Aerospace Engineering has a broad experience with different forms of automation in the design process. The department conducts amongst other research in the application of KBE and Multi-Disciplinary Optimisation (MDO) in different processes in the aerospace engineering industry.

1.4.2. KE-WORKS
KE-works is a Knowledge Engineering company specialised in the optimisation of engineering intensive projects in the manufacturing industry. KE-works optimises processes by the application of standardisation, knowledge re-use and automation. It is their aim to reduce project lead time and overall cost. KE-works developed a web-based WMS called KE-chain. KE-chain provides actors in the PDP with the required information in the right format and at the right time. This actor could be a human resource but also an external application. Hence the system supports for automation of activities in the process or automation of complete chains of activities defined by the user.

1.4.3. NOESIS SOLUTIONS
Noesis Solutions N.V. is a Belgian leadering technology provider in the markets of Process Integration and Automation, MDO, Uncertainty quantification and Robust Design [16]. The company provides an optimisation platform called Optimus, allowing for the application of different optimisation strategies and workflow simulations.

1.4.4. FOKKER TECHNOLOGIES
Fokker Technologies is a designer and producer of aerospace structures delivering to the major aerospace co-operations such as Airbus and Gulfstream. Fokker Aerostructures is the business unit specialised in lightweight structures and produces structures such as flaps, rudders but also complete tail sections. In this research the business unit Aerostructures is involved.

1.5. THESIS STRUCTURE
This first chapter has provided a brief introduction in which the need for this research has been identified and how this research is conducted. In the following chapter (Chapter 2) additional relevant background information is discussed. It focusses on the PDP and its performance, information quality, PDP modelling and analysis and automation in the design process. Chapter 3 then discusses the modelling approach as developed in this research. In the following chapters, Chapter 4 and Chapter 5, the simulation method and optimisation strategy are discussed in more detail respectively. The interaction and integration of the different elements (modelling framework, simulation and optimisation) is then discussed in Chapter 6. Verification and validation are of importance and therefore are discussed in Chapter 7. This verified framework is applied on a case study which is discussed in Chapter 8. Finally in Chapter 9 conclusions are drawn and a critical

view on the research is presented resulting in recommendations.

2 Background

In the field of Product Development Process (PDP) research, many definitions are used and multiple viewpoints on the same topic exist. The goal of this chapter is to provide a clear overview of these definitions and viewpoints and indicate how these are used in this research. Furthermore different concepts used in this research are explained.
First the PDP is discussed in Section 2.1. Next the performance of the PDP as defined in literature is discussed in Section 2.2, this is also important in determining the objective function in optimisation later on. Next, in Section 2.3, the quality of information is discussed since many automation initiatives aim at improving this quality. Following this the modelling and analysis of the PDP is discussed in Section 2.4. And finally in Section 2.5, multiple aspects of automation are discussed.

2.1. The product development process

The PDP is a broad term, applicable to many different processes for a wide variety of products and services. This section discusses the process and specific characteristics for complex aerospace engineering product.

2.1.1. General product development processes

In this report the PDP, challenges in the PDP and possible solutions are discussed. To discuss this, it is needed to have a clear definition of the PDP. The PDP is defined in literature in multiple ways. A selection of a few definitions is given below:

- "The transformation of a market opportunity and a set of assumptions about product technology into a product available for sale" [17].
- "Product Development (PD) is the process of transforming customer needs into an economically viable product that satisfies those needs." [18]
- "Product Development (PD) can be described as a complex web of interactions, some of which precipitate a cascade of rework among activities." [19]
- "Product Development is an endeavour comprised of the myriad, multi-functional activities done between defining a technology or market opportunity and starting production." [20]
- "PDP is a collection of activities which link the engineering techniques, methods, tools and people together and apply them into the product development practice. PDP involves technology and management issues." [21]
- "Product development is considered to be a process of transformation of input information about customer needs and market opportunities into output information which corresponds to manufacturable designs, and functional tooling for volume production." [22]

The definitions share some commonality: they all are a definition of a process implying a change in state. This process starts with an option for a new product (e.g. a new technology or need from a customer) and ends with a product or service that can be sold to the customer. Technology and/or activities are needed to facilitate this change in state.

The definition of the PDP used in this research is adopted from Krishnan and Ulrich [17]: ”The product development process is considered to be a process of transformation of input information about customer needs and market opportunities into output information which corresponds to manufacturable designs, and functional tooling for volume production.” The goal of the PDP is to deliver a product that meets all technical requirements, while keeping lead time and development cost as low as possible. Maximizing technical performance while minimising development costs and lead time is a great challenge, especially in case of complex product such as aircraft and aircraft systems. Here the number of interacting systems and involved disciplines is large, and the type and strength of such interaction is not always easy to grasp and manage effectively.

2.1.2. COMPLEX AEROSPACE PRODUCTS

A complex system is defined by van Tooren *et al.* as: "systems characterized by the fact that it is difficult, if not impossible, for one person to understand all the details of its subsystem and all the interactions between those subsystems. This complexity may be a result of large scale (e.g. many subsystems, very large numbers of variables), but also interactions between subsystems." [23] The complex product is thus a network of different components with technical interfaces to function as a whole, as illustrated in Figure 2.1.

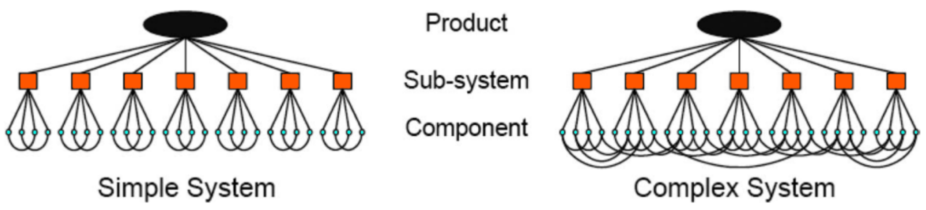


Figure 2.1: Illustration of simple and complex engineering systems [2]

The interactions between subsystems is not solely limited to the product itself. The interaction is also present between process steps in the PDP. Hence a complex process is inherent to a complex product.
This combination of a complex product and a complex process creates challenges in the controllability of the PDP. Managing the process by setting the right deadlines, budgets, resources, schedules, interactions etc. becomes extremely difficult in these cases. The effect of a single decision or change can become hard to predict in the complete PDP. Abdelsalam and Bao [24] state that additional challenges are imposed by the technical difficulty in complex engineering products but also the managerial complexity imposes difficulties to manage the interaction between different engineering disciplines.

2.1.3. PDP CHARACTERISTICS

In literature the PDP is often characterised by terms like ’creative’, ’iterative’, ’collaborative’ and ’innovative’ [14, 17, 19, 25, 26]. These PDP characteristics provide specific challenges which generally differ from those encountered, for example, in the manufacturing process. A selection of the characteristics relevant to this research is discussed in the following sub-sections.

ITERATION

The causes of iteration can differ; often a distinction is made between planned and unplanned iteration [3]. Planned iterations occur when a task is attempted without a complete set of information and hence assumptions are made that need to be verified later on. The concept of planned iteration is illustrated in Figure 2.2. First, the value of X is estimated and once it can be computed it is fed back and the planned iteration occurs.

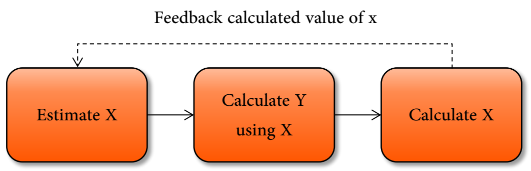


Figure 2.2: Example workflow demonstrating the principle of planned iteration

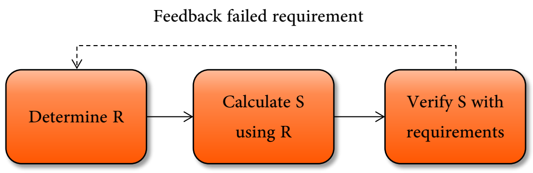


Figure 2.3: Example workflow demonstrating the principle of unplanned iteration

Unplanned iterations occur when activities are repeated due to unexpected failure, for example due to a (external) change in the requirements or by unexpected failing to meet a requirement. An example is illustrated in Figure 2.3. Here the value R (e.g. a design vector) is determined. Subsequently a value, S is calculated using this R and afterwards it is verified if S meets a set of requirements. If not al requirements are met, unplanned iteration is at hand and a different R should be determined.

REWORK

Repeating or refining a task (i.e. rework) is a consequence of iteration. In many cases, iteration has a second order effect in terms of rework: if one task changes many subsequent tasks need to be adjusted too. Literature discusses this effect extensively and methods are proposed to quantify the probability of rework in the case of a change, and the extent of rework necessary for the whole task (e.g. is it necessary to perform the full task again or only a selection of the task activities) [27].
Also the concept of the improvement curve is discussed in literature, meaning that the duration of a task decreases at each iteration performed by a human resource, due to the cumulated experience and increased ability of the resource [19]. Thus at every iteration the lead time is reduced by a predefined amount based on the applied relation of the improvement curve. Often this curve flattens out until a certain threshold level (i.e. the absolute minimum lead time with maximum amount of completions).

COLLABORATION

The PDP of complex engineering products is inherently a multidisciplinary process as discussed by Reed *et al.* [28] in the context of the aerospace industry. Multiple disciplines, often clustered in departments, need to interact and exchange information and trigger each other to start an activity. These collaborative activities have an influence on the performance of the PDP [29]. For example if one resource finishes a task it triggers a subsequent task by sending an email. In reality the resource responsible for the subsequent task is not constantly reading the incoming emails. Furthermore once the resource had read the email it does not mean that it will directly start working on the task. It could very well be that first another task needs to be finished. Hence this aspect of collaboration influences the total lead time of the process.

2.2. PERFORMANCE IN THE PRODUCT DEVELOPMENT PROCESS

To be able to optimise a process one needs to be able to measure the performance of the system. This performance can be stated in an objective function. This objective function can be a function of just one parameter or a combination of many (multivariate objective function). Such an objective function can also be created for the optimisation of the PDP. This optimisation is discussed more elaborately in Chapter 5.
Performance indicators are used to keep track of the systems performance and are of interest in this research. Often these performance indicators are abbreviated with Key Performance Indicator (KPI). As the word is already saying, it is a key indicator of the performance. Fitz-Gibon defines a performance indicator as "an item of information collected at regular intervals to track the performance of a system" [30, p. 1]. Examples of KPIs can be the number of clients per day of a barber, the weight of a designed product or the cost of a product.
This research focuses on improvement of the PDP. Therefore it is important to define a suitable set of performance indicators and look at strategies to improve them. In literature three common KPIs are based on time, cost and quality [7, 31–33]. Upon investigating these KPIs, a few interesting observations can be made. Firstly, most of the authors do not quantify the KPIs. This is in particular the case for product quality, which is often mentioned, but virtually never quantified to a measurable performance indicator during the PDP. Secondly, most authors focus on a single KPI. Even when multiple KPIs are addressed in one research, most of the optimisation studies are performed on a single KPI (i.e. no multi-objective optimisation). Since the previously mentioned KPIs are very broad the KPIs used in this research are discussed in more detail in the following paragraphs.

2.2.1. COST

In commercial processes cost is in virtually any case the most important indicator. It can be difficult to determine the actual cost since many cost attributes are not directly visible (e.g. the re-use of specific developed knowledge or systems).
In the PDP a distinction can be made between recurring and non-recurring costs. In this research the process costs is defined as the recurring cost. Process cost is the cost of the resource being occupied by a task and is a recurring cost in the process each time the task is executed. Investment cost is the total cost for an investment (e.g. to develop an automation solution for a certain process activity) and is a non-recurring cost. Costs such

as the cost of a license are also seen as an investment cost in this research although it is a yearly recurring cost.
The difficulty with non-recurring cost is that it often is an investment which can be discounted over multiple projects. The amount of projects is however often unknown at the start and therefore the cost of an investment is difficult to directly compare with a reduction in recurring cost.
The actual cost of the product under development in the PDP is not discussed in this research.

2.2.2. Time

In almost all literature on performance improvement of the PDP time is mentioned as a performance measure. It differs per source how this time is measured and what aspects are important. Some authors research a reduction in total lead time whilst others focus on methods to reduce a specific time attribute (e.g. waiting time).
Two relevant KPIs addressed in this research are lead time and process time. Lead time defines the total time from the beginning of the project until the end and consists of process time and waiting time [31]. Process time is defined as the time a resource is occupied by an activity over the course of a process. Other examples of performance indicators used in literature are waiting time [31, 34], iteration time and time schedule risk [19].

2.2.3. Quality

Product quality is an often mentioned indicator, but in most of the cases in PDP literature it is not defined, and with high exemption it is converted to a quantitative measurable indicator. Defining the product quality after it has been created has been done before with methods like Value Engineering and Earned Value Management. Defining the quality becomes more difficult if it needs to be measured during the process of development. In literature some examples can be found where quality is qualitatively discussed but is not used as a measurable indicator [32].
On this product quality alone also multiple views exist. The two most used views are the technical product performance and the customer product value. The technical product performance is the way an engineer traditionally refers to performance. It is the way in which the product meets the technical requirements and required functionalities. The other view is the customer product value and takes into account aspects like operating cost and on-time delivery.
For both ways of looking at product quality it should be noted that product quality is influenced by the development process. On the technical product quality level, the process influences product quality by having a limited amount of iterations or missing feedback loops. If the process has a longer lead time the product cost obviously increase and hence customer value decreases, hence the process can also influence the customer value.
A valid statement would be that the product quality increases with the amount of iterations made. This statement is supported by experts in industry. During an interview it was stated that more iteration loops would result in a better design (better was in this case lighter). The explanation for this increase in quality was that more iterations would yield more convergence and reduced margins leading to a lighter (in this case higher quality) product.
Although this topic is discussed in this background section, quality is not directly taken into account in this research.

2.3. Information Quality

The concept of Information Quality (IQ) is a relevant concept for this research. In the PDP literature, information is defined as the developed product and its corresponding information model. This information model is generated by multiple departments and/or resources. Hence the information is transferred throughout different process stages, department, resources and applications. This increases the need for high quality information, since otherwise it might cause processes to fail.
de Vrught [5] defined seven levers for IQ which are stated in Table 2.1.

Completeness and Correctness address the intrinsic IQ, whilst Currency and Relevance influence the contextual IQ. Accessibility and Traceability influence the acquisitional IQ and compliance is concerned with representational IQ. These levers are discussed in this section since different automation initiatives and levels of automation influence different IQ Levers.

Table 2.1: Overview of Information Quality Levers as defined by de Vrught [5]

Information Quality Lever	**Description**	**Information Quality topic**
Completeness	The extent to which the provided information is complete.	Intrinsic
Correctness	The extent to which the provided information is inherently correct.	Intrinsic
Currency	The extent to which the provided information is up to date and not obsolete.	Contextual
Accessibility	The extent to which the provided information can be retrieved continuously, autonomously and unobstructed.	Acquisitional
Traceability	The extent to which the provided information and the decision making process can be traced over time.	Acquisitional
Relevance	The extent to which the provided information contains information that is applicable for the execution of a succeeding process task.	Representational
Compliance	The extent to which the provided information complies with format requirements set by succeeding process tasks.	Representational

2.4. PDP MODELLING AND ANALYSIS

Modelling and analysing a process can be done in many different ways, a wide variety of process modelling languages and programs exist. According to van der Aalst and ter Hofstede [35] one of the reasons for this sprawl of models is the variety of ways in which business processes are described. This section first discusses modelling of the PDP in Section 2.4.1. Section 2.4.2 subsequently discusses methods to analyse it.

2.4.1. PDP MODELLING

A model is a simplified representation of the reality. "Ambiguities, uncertainties, and interdependencies among activities, their results, people and their tools make PD processes complex and challenging to model."[32] Many sources in literature propose modelling methods to model the PDP taking into account, among others, the characteristics mentioned in Section 2.1.3. All of these methods are based on the observation of specific behaviours. For example, models are proposed to account for the overlapping of processes [9], iterative loops [36] and the dynamic and stochastic aspects of the PDP [37].

Most PDP models use an activity network as a fundamental framework [32]. Here the process is viewed as a group of related activities that work together to create a result of value [38]. The PDP is a heterogeneous process, meaning that it consists of different types of activities, each having its own characteristics. Many process models do not make a distinction about the content of a task (i.e. a task is not decomposed into separate and different types of activities). For an extensive review on activity network-based process model, the author refers to a review by [32]. According to Browning the process architecture can be defined as the elements of process activities and their pattern of interaction [19]. This means that the process architecture not solely defines the elements of the activity itself, but also its interaction with the other activities.

A selection of frequently used modelling constructs is summarised in Table 2.2. In this research the Design Structure Matrix (DSM) is used to model the process and is therefore discussed in the following paragraphs.

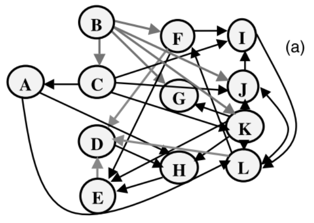
DESIGN STRUCTURE MATRIX

The DSM is a popular modelling tool. A DSM displays the relationship between components of a system in a compact, visual and analytically advantageous format [39]. A DSM has identical row and column entries and thus generates a square matrix. The elements on the diagonal have no purpose and are often blacked out in the standard version. The other off-diagonal items give information on the relation between system elements. Depending on the sign-convention the upper or lower triangle is the input or output. In this research

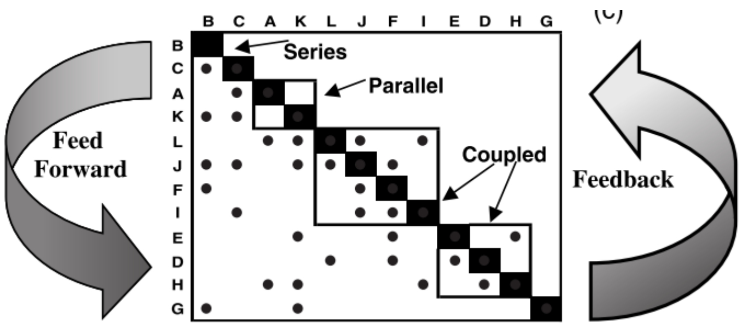
Table 2.2: Overview of different PDP modelling constructs used in literature

Modelling construct	**Description**
Information dependency	Stating the dependency between tasks based on their input and output relations. Often used in iteration and rework determination
Sequencing	Describing the precedence constraints between activities
Triggering and timing	Describing starting conditions for an activity based on for example a specific time (e.g. 8 o'clock)
Duration	Describing the duration of an activity
Cost	Used to model cost of resources, investments or revenue
Resources and capabilities	Modelling of specific resources with specific capabilities and properties
Variability	Used to model variability in for example duration or cost

the convention is used that marks below the diagonal are considered feed-forward and above the diagonal are considered to be feedback marks. Generally four types of DSMs are identified: parameter-based, activity-based, team-based and component based [3]. This research uses the activity-base DSM.
In Figure 2.4 the translation of a graph network of activities and information dependencies into an activity-based DSM is shown. In Figure 2.4a the graph network with information dependencies is shown but it is hard to extract the important information from this image. This graph network is translated into a DSM in Figure 2.4b, here the information dependencies can be seen clearly and also sequencing of activities is clear.



(a) Graph



(b) DSM

Figure 2.4: Overview of DSM (adjusted from [3])

As one can see in Figure 2.4b the DSM gives a very good overview of feedback loops and sequences of the process. Due to the matrix notation this model allows for integration with analysis methods as is discussed in the following section.

2.4.2. PDP ANALYSIS

A PDP model only visualises or formalises a process, but it lacks analysis of the model. It is often of interest what the performance of the constructed model is (e.g. lead time of the modelled process). This means that a user wants to evaluate the constructed model. Due to the complex interaction between activities, stochastic variables and many conditional aspects (e.g. availability of resources) mathematical analysis is often not possible. In this case simulation is a frequently used method for analysis.
Analysis by means of simulation is, for example, used to determine the total cycle time [24], the cost and schedule risk for various process architectures and to explore iteration and process structure [19].
Different types of simulation can be used depending on the modelling framework use to describe the process.

In the case of modelling by means of a DSM a discrete simulation method is preferred due to the discrete modelling of separate activities without continuous functions within these activities.

2.5. AUTOMATION

Automation can have a high impact on the performance of a process. Automation is a very broad term and it is applied in many different industries and processes [40, 41]. It is important to have a clear understanding of how to see automation in the PDP in this research. This section provides background information on automation.

2.5.1. DEFINITION OF AUTOMATION

Hart and Valasek [42] define automation as the ability of computer systems to perform a function without human support. Within the complex structure of any (PD)process, a number of tasks can be identified, each one implying the execution of a number of activities. In general, each one of these activities offers the opportunity to be executed with a different amount of human intervention. In other words, each activity offers the opportunity to implement a different level of automation. According to the author, design automation is about the process of transferring domain knowledge, in the broadest sense, from the expert to a computerized system, such that the system can systematically (re)use the captured knowledge to reduce, or eliminate the human involvement in some or all of the activities involved in the PDP. It appears that different levels of automation can be established, depending on the granularity level used to decompose a design process.

2.5.2. LEVELS OF AUTOMATION

Levels of automation have been researched in different application fields [42, 43]. Several models are proposed in literature, which differ, also significantly, in the identified number of automation levels and their granularity. For example, models are proposed with a number of automation level ranging from three [44] up to ten [45]; models exist that account for the different activities that are included in a task, whilst others see a task as one block, to which one level of automation can be assigned.

A general limitation of the level of automation metrics found in literature, also of those with higher granularity, is the inability to address the collaborative aspects in the PDP. No existing model, for example, defines levels of automation for typical PDP tasks such as “triggering next step”, “storing information”, “reporting”, etc.

Miller and Parasuraman [46] discusses that a task often can be decomposed to a lower level and that the level of automation on these levels can differ with respect to their parent task. They state that the previous framework of Parasuraman *et al.* [45] "does not go far enough". Miller and Parasuraman argue that automation may be applied differently to sub-tasks. They argue that "the profile of automation levels should stretch not merely over the four information processing phases at one level of decomposition, but over as many subtasks and levels as we want or need to divide a parent task into". Therefore they proposes to determine the level of automation for each sub-tasks until the level of detail needed for the problem at hand is reached. Miller and Parasuraman solely presents this finding but does not come up with a modelling framework solving the identified issue.

In conclusion, none of the level of automation models available in literature was deemed suitable for the purpose of this research; thereby a new one was devised, which is elaborated in detail in Section 3.3.

2.5.3. TYPES OF AUTOMATION

Not all automation initiatives are alike. Automation initiatives differentiate from one another in the core of what type of activity it automates.

INFORMATION FLOW AUTOMATION

In complex PDPs different departments are involved, often at different physical locations. These departments interact by exchanging information. To be able to work efficiently and effectively it is needed that this information is available in the right place, at the right time and in the right format [19]. Research by Brandao and Wynn [26] estimated that 30% of the development time is spent on searching and interpreting the information. This already shows that there is a lot of potential for reducing waste in this information flow by applying automation [13]. Automation in this case does not add direct value to the development of the information model of the product at hand, but improves the information flow throughout the process. Referring back to the discussion on Information Quality in Section 2.3, the automation of the information flow is about improving the contextual and acquisitional IQ. Some examples of information flow automation are knowledge capture and re-use, product and process management and providing communication and collaboration tools [26].

INFORMATION VALUE AUTOMATION

The information value automation concerns the automation of the activities directly adding value to the information model of the product under development. This means it adds information about the product itself (e.g. the strength of the bolt) but it can also be the activity of determining the settings for a specific analysis. In the terms of IQ the intrinsic IQ is improved with this type of automation.
Two well known methods for automation in PDP are Design Automation (DA) and Knowledge Based Engineering (KBE). van der Velden *et al.* [40] discuss the difference between the two technologies. However the goal of both technologies is similar: improve the PDP by using (information) technology. van der Velden *et al.* [40] point out the difference to be the scope of application. Where KBE applications are viewed as an integrated systems solution, Design Automation applications tend to be stand-alone applications to support the process. Since this research investigates the automation of a PDP by looking at the individual activities in the process, the term Design Automation is adopted throughout this research when referring to the process of information value automation.

2.5.4. EFFECTS OF AUTOMATION

Automation potentially has a large effect on the performance of the PDP. The application of automation generally increases non-recurring cost (investment cost in this research) and reduces the recurring cost (process cost in this research). Automation often leads to a reduced activity lead time and improved product quality. van der Velden *et al.* [40] discusses other intangible benefits of the application of automation:

- Consistency: Dedicated tools with standardised inputs and outputs can provide greater consistency.
- Complexity: Simplification of standardisation of complex processes, minimising possibility for human error.
- Integration: Custom automation tools can interface directly with existing engineering software, proving seamless automation and assurance that outputs are derived from previously validated methods and software.
- Change management: Ability to regenerate results as changes are made throughout the project.

However, also risks exist in the application of automation. For example, automation can negatively influence the complacency of the engineer and the situational awareness of the engineer [47]. Examples of automation complacency can be found in literature and reality, an example is the grounding of a cruise ship near the coast of Nantucket. The accident was caused by a failure of satellite based automatic navigation and because the crew did not monitor any other source of navigation [48]. Furthermore automation provides an easy option to generate a design and does not force the designer to think creatively for alternative solutions, hence reducing innovative and creative ideas. Although these effects are hard, if not impossible, to predict and quantify, they are important to take into account upon trading off alternative levels of automation.

2.5.5. POTENTIAL FOR AUTOMATION APPLICATION

Not all processes are as suitable as others for the application of automation. Multiple authors have suggested different criteria and techniques to assess the suitability of a process (on different levels of granularity) for automation [13, 40, 49]. Both the technical potential for automation and the cultural environment in a company should be taken into account in making a decision on the application of automation.
A process has a certain technical automation potential based on multiple criteria. Emberey *et al.* [49] takes into account if a process step is a routine process step, if it is based on formalised rules and if it is a complex step (i.e. involving multiple independent interacting steps). van der Velden *et al.* [40] discusses that low level, repetitive tasks are suitable for automation. Furthermore criteria like the integration with external application, amount of reporting and standardisation are indicators of suitability for automation. These factors influence the cost of automation of an activity in a process and should be taken into account in this research.
Besides these technical aspects, also the cultural aspects are important for a successful application of automation in a process. van der Velden *et al.* [40] lists multiple cultural aspects, of which a selection is discussed here. A cultural aspect could be the perception of the human resource on the automation initiative

as a threat to its job security. Furthermore experts acknowledge that, in some cases, an engineer is comfortable with the current way of working and is unwilling to change. An organisation might be unwilling to use automation due to (amongst others) the risks as mentioned in the Section 2.5.4. A company often is also reluctant in contracting a specific automation company due to the lock-in effect with the companies proprietary software. The last cultural aspect mentioned by van der Velden *et al.* [40] is that the application of automation might potentially damage the relationship with suppliers.
These cultural aspects are often applicable on a organisational level and not specific for a certain task or activity.

3

PDP MODELLING APPROACH

This chapter discusses the Product Development Process (PDP) modelling approach used in this research, taking into account the background as discussed in the previous chapter. The proposed approach is a core element of this research since it describes how to model the PDP and its behaviour. Furthermore this approach suggests novel methods to address the effect of different levels of automation on PDP performance. First the modelling approach philosophy is stated, this should provide more insight into why this modelling approach is developed in this proposed way. Subsequently the individual constructs of the approach are discussed. These constructs can roughly be subdivided in process modelling (Section 3.2), levels of automation modelling (Section 3.3), activity and process lead time estimation methods (Sections 3.4 and 3.5) and cost estimation methods (Sections 3.6 and 3.7).

3.1. PHILOSOPHY

As stated in Chapter 1, currently a trade-off between lead-time and development cost on incremental levels automation is rarely made. Furthermore the available PDP frameworks are not well equipped to estimate the effect incremental automation. Since this is an important aspect in this research, it is of importance that the proposed approach is able to model this effect of an incremental application of automation.
In a process multiple tasks are executed and within these tasks multiple different activities can be identified. These activities within tasks are unlike in their potential lead time reduction and the accompanied cost for automation. For example the activities *querying a database* and *analyses of structural component* can differ in both lead time reduction and cost of automation. The modelling approach should take into account this behaviour and be able to identify these different activities and their implications on the process performance. The aspect of modelling these different tasks and activities is discussed elaborately in Section 3.2.
As discussed in Chapter 2, multiple levels of automation can be identified. These different levels of automation have different effects on the process performance. This is taken into account in this modelling approach and it is discussed in Section 3.3. One of the objectives of this research is to develop a method to provide insight to decision makers in the costs and benefits of the application of automation. Therefore the model should be able to estimate both the cost of automation of an activity and the difference in lead time due to the different levels of automation. These methods are discussed in sections 3.4, 3.6 and 3.7.

3.2. PROCESS MODELLING

As stated in Section 3.1 the process needs to be modelled at a specific level of granularity. By modelling a process on an activity level it is possible to assess the effect of automation on different activities in the PDP. As was concluded in Chapter 2 the assessed models used in literature do not meet the requirements for this research and hence a process modelling method is proposed in this section. This method comprises two important aspects, the process granularity and the modelled activity types.

3.2.1. PROCESS GRANULARITY

A process can be modelled at different levels of granularity. The structure proposed in this research is illustrated by the example in Figure 3.1. Since the goal is to investigate the effect of automation on a generic process, the model decomposes any specific process in identifiable specific tasks (the top two levels in Figure 3.1), and, finally, each specific task into a set of generic activities. These activities can be seen as the building blocks of any PDP.

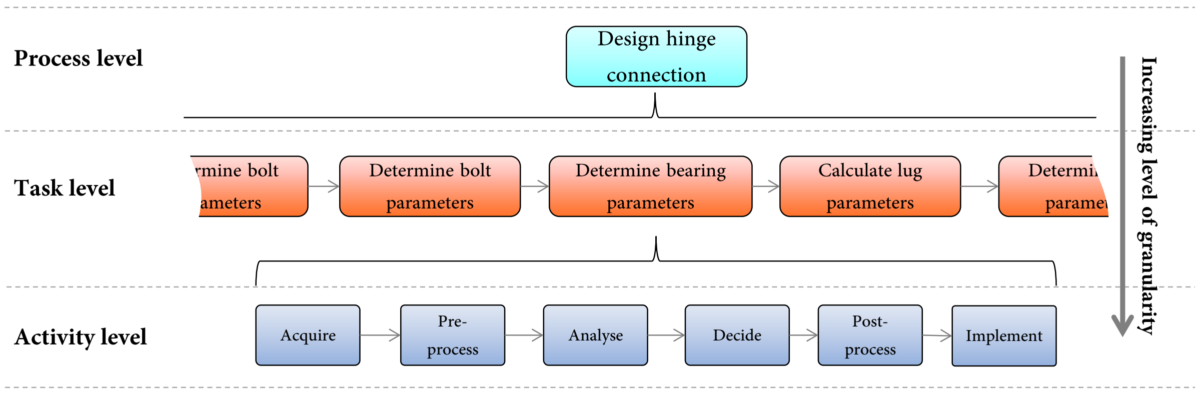


Figure 3.1: Example of process decomposition in tasks and activities

The granularity of the tasks still can be ambiguous. If one would for example define a task like *'Design aircraft landing gear'* the level of granularity would be too big to be able to directly decompose it into the level of granularity of the proposed activities (see Figure 3.1). Therefore requirements are used to test whether or not a process is decomposed in tasks with the right level of granularity. These requirements are:

1. Task is executed by one (set of) resource(s). This set should be involved with the task over the full duration of the task.
2. Task uses one system/device/platform. Meaning no interfaces with multiple platforms. If information is transferred between different platforms then this task is not defined at an appropriate level and should be decomposed in separate tasks.
3. Task can be executed without interruptions. Meaning that no additional information or waiting is required during the full duration of the task.

If a task meets these requirements it is defined at the required level of granularity and can be decomposed into the activities. These activities will be discussed in the following section.

3.2.2. ACTIVITY TYPES

Based on extensive literature research and investigation of four industrial cases, five activities were selected (from now on simply referred to as activities) as essential building blocks for any process:

- Acquire
- (Pre/Post-)Process
- Analyse
- Decide
- Implement

A comprehensive definition of these five activities is given in Table 3.1. It should be noted that no fixed sequence or amount of activities is prescribed for a task. As examples, one could model the Task "determining bearing parameters" in Figure 3.1 as a combination of the activities Acquire, Decide and Implement. The Task "calculate lug parameters" could be modelled as a combination of the activities Pre-process, Analyse and Post-process, where both the Pre and Post-process activities are of the same 'Process' type activity described in Table 3.1.
In practice, any process of any type of complexity can be split into tasks and eventually modelled as collections of these five predefined activities. A task can be defined as a sequence of (some or all of) these activities, with their relative duration distribution in the given task. The sequence is not predetermined, hence it can be the case that a decision activity type is followed by an analysis activity type. Two example task decompositions are given in Figures 3.2 and 3.3. The duration of the acquire activity of task X is defined as 0.2 (20 %)

Table 3.1: Comprehensive overview of activity characteristics

Activity type	Description	Examples
Acquire	This type of activity is concerned with acquiring all the starting conditions for a subsequent task from an external source. A starting condition is for example a trigger, knowledge or a physical product. These starting conditions are not transformed in anyway, it is acquired in the raw format as it is available.	Retrieving the database with all available materials and their properties from the shared environment. Acquire email to start working on stress analysis.
(pre/post) Process	This activity structures the information and represents in such a way to improve the relevance of the information. Processing information can also be applied to improve the relevance or compliance of the information. In this task no information is added to the product model other than transforming units. Structuring of information can also be done with the output of an activity to generate a specific report for example.	Filtering the database of all materials for all materials within budget constraints. Generate margin of safety report.
Analysis	In an analysis activity information is transformed and new information is created. This information is added to the information model. Knowledge is used to transform the inputs to outputs.	Calculate bolt strength for cheapest material type. Simulate process. Perform FEM analysis. Model product in CATIA.
Decide	This activity is a gateway where a decision is made with an impact on the process. At least two alternatives should be present for a decision. In this task no information is added to the product model.	Verify if bolt strength meets requirements. Determine hinge type philosophy.
Implement	This activity accounts for all the interaction with external (re)sources required to successfully continue the process. No new information is created but it is stored at a location. This task also accounts for triggering the next task.	Store stress report in shared environment. Send email to colleague to order new bolts.

of the Total Duration (TD) of the task. Similarly the analysis and implement task have a duration of 55% and 20% of TD respectively.

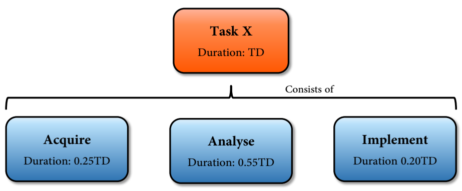


Figure 3.2: Example task X decomposed in activities with corresponding durations

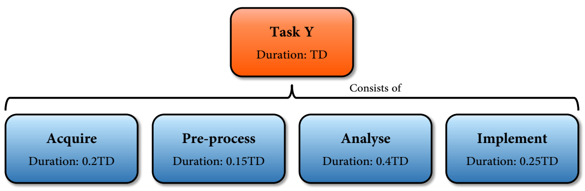


Figure 3.3: Example task Y decomposed in activities with corresponding durations

For this methodology it is essential to correctly classify the activities in the task to match with the activities in the framework. Therefore for every activity a few questions can be asked to be able to classify them as the right type of activity. Based on the following interview questions the activities can be identified to be of a certain type:

1. Does this activity determine new information to be added to the information model? *If yes, this activity can be an analysis or decide activity. If no, it can be an acquire, structure or implement activity.*

   Does this activity influence the process flow (i.e. does it potentially lead to rework)? *If yes, this activity should be split up in an analysis and decide activity. If no, the activity is an analyse activity*

   Does this activity require information, tools or methods? *If yes, an acquire activity is required to precede this activity.*

   Does this activity require implementation of the new information and corresponding activities? *If yes, an implement activity is required to succeed this activity.*

2. Is the data structure of the information altered in this activity without adding new information? *If no information is added and only information is altered in terms of the structure then the activity is of the type structure*

3. Is information added to the working memory in this activity or service? *If information is acquired from a database and is to be used in subsequent activities the activity type is acquire*

4. Is information implemented in the information model? *If information is implemented in the information model the activity is of the implement type*

These interview questions only serve as a guide and are not complete. In the processes defined within companies often the steps of acquiring the information, getting triggered and structuring the information is not stated formally. For this methodology this is however an important aspect to explicitly state in the process to take into account reductions in lead time due to this automation in the information flow. For the activities that are identified also their pre- and post processing needs to be added to the process model.

3.2.3. ACTIVITY INTERDEPENDENCY

The interdependency between activities is an essential aspect in modelling of the PDP. As discussed in Chapter 2 the sequencing of the process influences the process performance to a large extent. Furthermore this interdependency represents the actual case of the PDP in which value is created by receiving, processing and producing information.
In this framework the activity-based Design Structure Matrix (DSM) (see Section 2.4.1) is used since it is a powerful method to represent the relationship between activities. Due to its format, as can be seen in Figure 3.4, the difference between feed-forward and feedback relations can be identified easily and hence it quickly provides insight in the process.

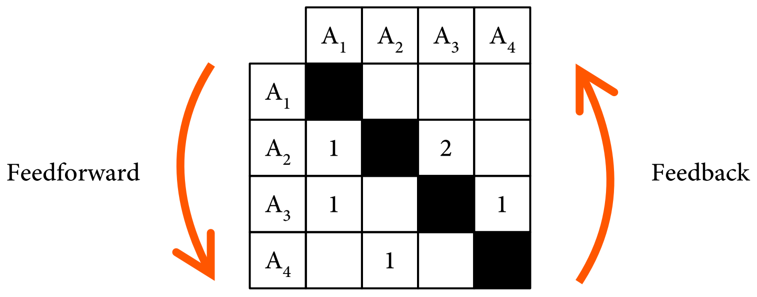


Figure 3.4: Basic example of an activity-based DSM

Use is made of a numerical DSM in which the off-diagonal numerical values above the diagonal represent the amount of iterations. The amount of iterations is the number of times the activity provides feedback to the row in which the value is stated. A numerical value below the diagonal other than 1 does not have a specific meaning. A value below the diagonal only indicates that there is a feed-forward relation but any value other than 1 would not have a meaning.

3.3. LEVELS OF AUTOMATION MODELLING

This section discusses the levels of automation of the activity types as stated in Table 3.1.
Considering the current available scales of levels of automation discussed in Section 2.5.2, they fail to fulfil the needs of this research due either to a continuous scale or a scale with too many discrete steps to keep it manageable in the methodology of this research. Another issue with existing models is their inability to describe a complex PDP. By examining the models mentioned before it can be seen that most models lack the activities concerning the collaboration in the process and are focussed on human decision making not taking into account a collaborative environment.
Due to these issues it is decided to develop a custom framework to model the levels of automation to be able to assess the current level of automation of process activities.
For the development of the scale of levels of automation model, three principles were taken into account:

1. The steps require to be discrete steps
2. The amount of levels should be manageable and is therefore capped of at 4
3. The lowest level always should be the fully human (i.e. non-automated) level and the highest would be a fully automated task (i.e. no human involvement).

With these principles in mind a variety of automation initiatives are plotted on a grid. On the horizontal axis the activity types are stated and the vertical axis denotes a continuous scale for the level of automation. This results in the overview as displayed in Figure 3.5. The automation initiatives plotted on this grid are based on the experiences of KE-works and automation initiatives discussed in literature.

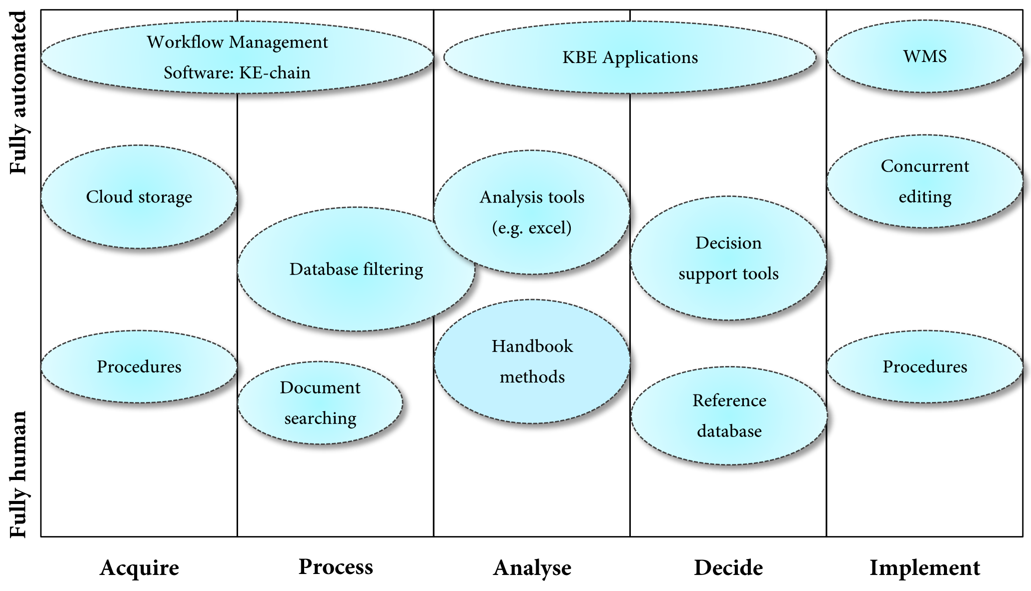


Figure 3.5: Grid with various automation initiatives plotted

Based on the grid of Figure 3.5 and the three principles stated in the previous paragraph the levels of automation are discretised. It is inherent to a selective amount of discrete steps that some initiatives are on the boundary between two discrete steps. This leads to some friction when applying a framework, the same principle holds for this framework. In this research, criteria have been developed to provide means to help to determine the level of automation of an activity, these criteria are described in Table 3.3.
The proposed framework is summarised in Table 3.2. For each one of the activity types defined in Section 3.2, four levels of automation are proposed, ranging from level 1, in which the human is the sole resource, to level 4, where full automation is provided by a computer. On the basis of the definitions provided in Table 3.2, each task owner in the PDP process should be able to describe the current level of automation, hence the type of resources involved in the execution of the encompassed activities.

Table 3.2: Summary of levels of automation (LoA) for PDP activity types

LoA	**Acquire**	**Process**	**Evaluate**	**Decide**	**Implement**
4	The system is the sole resource and automatically executes the activity and acquires the required items. The system is able to acquire information from difference sources, hence it is able to integrate with external software to gather information.	The computer is responsible to structure the information in such a way that the next activity accepts it to be in the right format.	Computer is fully responsible for this activity. Hence it is able to interpret the provided information and determine how to execute this activity successfully.	The computer decides and acts autonomously without interference of the human.	The computer is responsible for the correct execution of the implementation activity.
3	The activity is defined and the system suggests what to acquire and where it can be acquired. The source is responsible to acquire the items from the source. All information is always available from a single source of truth.	The computer supports the user in processing the information. Hence the knowledge for processing the information is in the system but the resource needs to decide on how to apply this knowledge (i.e. no automatic execution).	Computer supports the execution of the activity by providing tools and methods to perform calculations. Human interaction is still needed to determine intermediate steps or to verify the result.	Human is assisted by the system which interprets the data and shows and ranks all alternatives. Possibly the system suggests the best alternative for specific performance measures.	The human executes the implementation activity. The computer system supports the human and provides information on what to do and how to do it. System is actively involved by preventing certain actions or promoting others.

Continued on next page

Table 3.2 – *Continued from previous page*

LoA	Acquire	Process	Evaluate	Decide	Implement
2	The activity is defined and the system suggests what items need to be acquired and where to find them. The resource is responsible to acquire the items from the source.	The human is responsible to process the information. It is defined how to process the information for example by using templates.	The human is responsible for this activity and is assisted by handbook methods and procedures. Hence part of the knowledge base is in the handbook methods and procedures. The human remains the main source for the analysis.	Human is still responsible but the system assists in interpretation of the information and shows all relevant alternatives.	Human is responsible for the implementation activity but is supported by the system. System provides relevant information on who to contact and where to store information for example.
1	The human is the sole resource for the activity. Hence no assistance is provided by a system, manuals or procedures.	The human is responsible for processing the information. A computer or other system with basic features can be used to enhance information relevance.	Human is the only source for the methods and knowledge used in this task.	The human is responsible for the decision and the system does not provide assistance.	Human is fully responsible for the implementation activity. No assistance offered by the computer.

Different criteria are available to determine the current level of automation as discussed in Section 3.3. The information in Table 3.3 provides a means for the user (of the methodology) to determine the level of automation. The level of automation of an activity is determined based on compliance with specific criteria. These criteria differ per activity type and can be found in Table 3.3. It should be noted that this is not an exclusive list but it serves as guidance in determining the level of automation. The expert judgement of a knowledge engineer remains more valuable than these specific requirements.

Table 3.3: Criteria to assist in the identification of the levels of automation per activity type

LoA	Acquire	Process	Analyse	Decide	Implement
4	Is the activity automatically started? Is no interaction with the system required? Is the information acquired from single source of truth? Is the activity able to integrate with other applications? Is the subsequent activity requiring the information automatically triggered?	Is all information automatically processed by the system? Is the output of this activity in the predefined currency? Is the output of this step of a higher level of relevance or compliance?	Is the system able to automatically transform the inputs to the correct outputs? Does the system operate autonomously? Can the system automatically start without user interference?	Is the system responsible to make a decision? Is this decision not presented to the user for verification?	Is the system responsible for correct implementation of the preceding task? Does the system implement potential information in the dedicated location? Is versioning of implemented information available? Does the system signal the subsequent activities and resources?
3	Does the system provide information on what information is required for the subsequent activity? Does the system provide information on where to acquire the information? Is a single source of truth available?	Is a system available to process the information of this task by means of specific user inputs? Is the system developed specifically for this activity? Is information available on how to process the information? Is information available on requirements of the outputs of this task?	Is the system unable to automatically transform the inputs to the correct outputs? Is the majority of the knowledge and rules captured in a system and applied in a framework? Is minor user interaction required to interface and make small adjustments for example?	Is the user responsible to make a decision? Does the system interpret the information and rank different alternatives? Does the system interpret the information?	Is the user supported by the system in determining what to implement and where to do this? Does the system provide means to ensure correct implementation? Is a single source of truth available?
2	Does the system provide information on what information is required? Does the system provide information on where to acquire the information?	Is a system available to assist the user to process the information? Is information available on how to process this information? Is information available on the required outputs of this system?	Is a system available with information on how to perform the analysis (e.g. handbooks or guides)? Is the user responsible for the application of these knowledge rules?	Is the human responsible to make the decision? Is the human supported by tools to interpret the information?	Is the user supported by information on the activities to undertake to ensure correct implementation? Is information available on what the subsequent activities are?

Continued on next page

Table 3.3 – *Continued from previous page*

LoA	Acquire	Process	Analyse	Decide	Implement
1	Is the human the sole source to find the information? Is no system available with information on what information is required? Is the information on how to perform this activity merely tacit knowledge of the human?	Is the human the sole resource to process the information? Is no information available on the output requirements? Is the information on how to perform this activity merely tacit knowledge of the human?	Is no information documented on how to perform the analysis? Is no system available in this process to execute the analysis? Is the human responsible for transferring the inputs to the outputs? Is the information on how to perform this activity merely tacit knowledge of the human?	Is the human the sole source to make a decision? Is the human unassisted in making this decision with tools or method? Is the information on how to perform this activity merely tacit knowledge of the human?	Is the human the sole source to perform this activity? Is the information on how to perform this activity merely tacit knowledge of the human?

3.4. ACTIVITY DURATION ESTIMATION METHOD

The influence of the level of automation on the activity lead time can differ per task and activity type. This section discusses the method used to estimate the activity lead time for a specified level of automation.
This PDP modelling approach uses deterministic values and hence the activity lead time is a deterministic value, known for the current situation (i.e. in current conditions at the current level of automation). The purpose is to estimate the consequence on a given activity duration caused by a change of the level of automation (e.g. going from fully human to fully automated). To estimate this value a method is proposed which assumes a predetermined reduction in activity lead time taking into account:

- Current activity lead time
- Activity type
- Current level of automation
- Proposed level of automation

This predetermined reduction is captured in a coefficient. This coefficient represents the percentage of the time a task would take, measured with respect to the time of the given activity at a level of automation equal to 1.
The coefficients are given in the Duration Matrix, DM_{ij}, and used in determining the estimated activity duration at another level of automation. In this matrix the subscript i is the task activity type (e.g. acquire) and subscript j is the level of automation. A sample DM is shown in Table 3.4. Here it can be seen that all activity types are at 100 % of their duration if they are at a LoA of 1. For higher levels of automation their relative reduction in lead time differ per activity type and per level of automation. Hence it is possible that for an *acquire* type of activity a step from LoA 1 to LoA 2 has a relatively smaller effect on lead time than for a *decide* activity.

Table 3.4: Example of Duration Matrix (DM) for a fictitious task

	Level of Automation			
	1	**2**	**3**	**4**
Acquire	100%	80%	50%	10%
Process	100%	70%	40%	20%
Analyse	100%	90%	60%	15%
Decide	100%	65%	40%	30%
Implement	100%	60%	30%	5%

The activity lead time is calculated by using Equation 3.1. In this equation t^* and j^* indicate the estimated time and proposed level of automation respectively. The parameter t is the current activity lead time.

$$t^* = \frac{DM_{ij^*} \cdot t}{DM_{ij}} \tag{3.1}$$

The values of DM_{ij} are essential and influence the estimated activity lead time reduction to large extent. A correct determination of the coefficients is essential to the prediction capability of the proposed method. It

will be crucial for any company that is willing to adopt the proposed method to properly estimate such values and continuously improve and update them, based on internal project knowledge. Furthermore the values of the DM might differ per project, company or industry. The sensitivity of the values of the DM is discussed in more detail in Chapter 7.
To determine the values of the DM information is required on how the different activity types in the process behave at varying levels of automation. The method used in this research to determine these values is by performing a dedicated workshop (specifications of the workshop as executed in this research can be found in Section 8.2). In this workshop current and future users of the process under consideration execute a demonstration project multiple times. It is executed multiple times with varying levels of automation for activities. The project consists of tasks with similar challenges as the tasks in the process under consideration. Of these tasks the individual activities need to be monitored, Protocol Analysis (PA) as discussed by Milton [50] is advised.
Furthermore it is advised to select the response groups carefully, a response group of experts in the process under consideration are likely to yield different coefficients than a group of employees unfamiliar with the process. Also the process learning curve could influence the coefficients. If the same process is repeated with varying levels of automation the respondents become familiar with the process and that is likely to affect the lead time of the activities. Averaging between groups or averaging over multiple repeated identical experiments could compensate for this effect.
In Chapter 2 the effect of the learning curve was discussed briefly. For this modelling approach it would imply that the activity duration would also be a function of a learning curve factor and the amount of times an activity has been executed. The method as implemented in this research does not take into account this effect of the learning curve. Adjusting for the Learning Curve is recommended for future research however.

3.5. PROCESS LEAD TIME ESTIMATION METHOD

Calculation of the total process lead time is based on the duration of all activities. A sum of all activity lead times would however not yield valid results. Processes running in parallel, iterations, interruptions and other aspects need to be taken into account. Taking into account these aspects, it is not possible to conceive a mathematical method to estimate the process lead time. Therefore simulation is required, as will be discussed in Chapter 4.
In this simulation the process lead time is computed. In short, the lead time is equal to the difference between the starting time and ending time of the simulated process. In that specific simulation, the above mentioned aspects are taken into account. Hence if activities run parallel then the activity with the longest lead time determines the process lead time. Furthermore it takes into account waiting time induced by limited available resources, additional waiting time due to collaboration burdens (see Section 4.3.3) and other aspects which will be discussed in Chapter 4.
A limiting factor in the calculation of the lead time is that no working hours are taken into account. Meaning that all human resources are simulated to work 24 hours per day. It would be a valuable addition to adjust the lead time estimation for this effect.

3.6. PROCESS COST ESTIMATION METHOD

Besides the activity duration the process cost is an important aspect in the determination of the benefits of the application of automation in a PDP. By increasing automation, the process time of a resource is reduced or potentially omitted leading to reduced process cost of the human resource. This section elaborates on the method used to estimate the process cost.
The activity process cost of an activity is based on the type of involved resources, the cost of these involved resources and the activity lead time. The determination of process cost based on the estimated activity lead time is computed with equation 3.2.

$$A_C = [j < LoA_{max}] \sum_{i \in \mathbf{R}} r_i \cdot t \cdot \tag{3.2}$$

Where: A_C is the activity process cost, reflecting the cost if the activity is executed for the duration of t. t is the duration, usually equal to the activity lead time but in the case of an interrupted task this could be lower than the activity lead time. j is the level of automation. $\mathbf{R}$ is the list of involved resources. r_i is the hour rate of a the resource. If multiple resources are involved in an activity the sum of the cost per resource determines the total activity process cost, the summation in equation 3.2 accounts for this. The activity process cost per

resource for a specific duration is calculated by multiplying the resource hour rate, r_i with the duration t.
In activities without human interaction (i.e. level 4 of automation) the resources are not utilised and hence the activity process costs are assumed to be zero. Equation 3.2 accounts for this by the term between Iverson brackets. Here j is the level of automation. If the term in between the Iverson brackets is true it denotes the value 1, if it is false it denotes the value 0.
The total process cost is calculated by using equation 3.3. Here P_C is the total process cost. This total process cost is the sum of all activity process costs of all performed activities in the complete process lead time. In this equation $\mathbf{x}$ denotes the set of activity costs of all performed activities in the process (e.g. $\mathbf{x} = [A_{1,C}, A_{2,C}, A_{3,C}]$ for a process with three activities).

$$P_C = \sum_{i \in \mathbf{x}} A_{i,C} \tag{3.3}$$

It should be noted that costs such as Workflow Management Software (WMS) licenses are considered to be an investment and are accounted for in the investment cost.
The use of automation also enables the use of different (e.g. cheaper) resources. This effect of automation is not taken into account in the determination of the process cost. It is assumed that for the three lowest levels of automation identical resources are used, and hence resource hour rates remain constant. It is expected that this results in conservative process cost estimations.

3.7. AUTOMATION INVESTMENT COST ESTIMATION METHOD

Another metric of importance in this methodology is the investment cost required to automate an activity to the level of automation as stated in the design vector. In order to provide a meaningful estimation of the required investment cost, it is necessary to take into account the current level of automation and the type of activity to be automated. This section discusses the implemented cost estimation method.

3.7.1. ACTIVITY AUTOMATION COST ATTRIBUTES

The current cost estimation technique internally used by KE-works for knowledge engineering business is therefore adopted and modified for this research. This technique uses a roll-up technique combined with parametric relations based on empirical data and expert judgement. In this roll-up technique the different exercises required in the development of automation initiatives are identified. The exercises included in this roll-up technique are:

- Knowledge acquisition
- Application development
- Integration
- Configuration
- WMS license
- Server
- Training
- Management

For these different activities a cost estimation relationship is developed based on empirical data and expert judgement. Depending on the task type and level of automation the investment cost is estimated for each individual activity. In the following paragraphs these exercises are discussed individually.

Knowledge acquisition cost

The knowledge acquisition cost is split into the internal and external knowledge acquisition cost. The internal knowledge acquisition cost represents the cost of knowledge acquisition for the company conducting the knowledge acquisition. The external knowledge acquisition cost represents the cost for the company requiring the knowledge acquisition.
The internal knowledge acquisition cost comprises all activities in the full knowledge acquisition process. It takes for example into account the preparation, document reviews, interviews, knowledge modelling, teach-back and building a knowledge base. Equation 3.4 displays the formula to estimate the internal knowledge acquisition cost.

$$C_{KA_{int_A}} = \bar{t} \cdot r_{KA} \cdot k_{KA_{int}} \cdot (CM_{KA_{ij^*}} - CM_{KA_{ij}}) \tag{3.4}$$

Where $C_{KA_{int_A}}$ is the cost of knowledge acquisition per activity. In this equation 4 terms can be identified which are multiplied with each other. The first term is the normalised duration: $\bar{t}$. This is the lead time normalised to the lead time if the task would be at a level of automation 1. The lead time of the activity is normalised by using equation 3.5. The second term is, r_{KA}, represents the hour rate of the knowledge engineer performing the knowledge acquisition. The third term, $K_{KA_{int}}$, is a coefficient representing the relation between the activity lead time of an activity and the expected duration of the knowledge acquisition. Based on expert interviews and empirical data this relationship was determined. Due to confidentiality the value of this coefficient can not be disclosed. The fourth and last term is the term accounting for the effort required in the knowledge acquisition and the effort already invested in the available knowledge base. These values are extracted from the Cost Matrix for knowledge acquisition (CM_{KA}). This matrix provides coefficients for the effort required to automate an activity type to a specific level of automation. A sample CM_{KA} can be seen in Table 3.5. The first part of the last term in equation 3.4, $CM_{KA_{ij^*}}$, is the percentage of knowledge acquisition needed for the proposed level of automation (j^*) and the second part ($CM_{KA_{ij}}$) represents the percentage of knowledge acquisition already performed since the activity has already been automated to the current level of automation (j).

$$\bar{t} = \frac{t \cdot DM_{i1}}{DM_{ij}} \tag{3.5}$$

Table 3.5: Sample knowledge acquisition effort coefficients (CM_{KA})

	Level of Automation			
	1	**2**	**3**	**4**
Acquire	0%	30%	70%	100%
Process	0%	40%	80%	100%
Analyse	0%	45%	90%	100%
Decide	0%	30%	60%	100%
Implement	0%	30%	70%	100%

For the external cost of knowledge acquisition the cost of the time required from the domain expert(s) for interviews and validation is taken into account. Milton [50] provides a ratio between the internal knowledge acquisition effort and the effort from domain experts. This ratio is represented by $K_{KA_{ext}}$ in equation 3.6.

$$C_{KA_{ext_A}} = \bar{t} \cdot r_{KA} \cdot k_{KA_{ext}} \cdot (CM_{KA_{ij^*}} - CM_{KA_{ij}}) \tag{3.6}$$

The sum of the internal and external knowledge acquisition cost provide the total cost of knowledge acquisition as can be seen in Equation 3.7.

$$C_{KA_A} = C_{KA_{int_A}} + C_{KA_{ext_A}} \tag{3.7}$$

Development cost

The development cost is a complex cost to determine. Many different cost estimation methods solely for the development of IT solutions have been proposed in the past decades. As stated before, these models do not suit the needs of this research due to their applicability on large projects. In this research the development cost takes into account multiple sub-activities involved in development. In general the software development process can be broken down into the following sub-activities:

- Define system feasibility
- Create software plans and requirements
- Develop product design
- Develop product detailed design
- Coding
- Integration

- Implementation
- Operate and manage

All activities except for the last are accounted for in the cost attribute of development cost.
Generating a parametric model to estimate the cost of development is a challenge if one does not have detailed information on the activity itself. Therefore this research proposes to use expert judgement to estimate the cost of development. The formula used in this research to estimate the cost of development (C_{dev_A}) can be seen in equation 3.8.

$$C_{dev_A} = E \cdot r_{dev} \cdot K_{dev} \cdot (CM_{dev_{ij*}} - CM_{dev_{ij}}) \tag{3.8}$$

Where E is the required development effort in hours. r_{dev} is the hour rate of the developer. K_{dev} is used as a multiplier for other development activities and $CM_{dev_{ij*}}$ and $CM_{dev_{ij}}$ are the Cost Matrices for development. In equation 3.8, E, is estimated by the knowledge engineer. The knowledge engineer estimates the total amount of hours required to perform the coding sub-activity. The total amount of hours spend on all sub-activities is calculated by multiplying this estimate with a multiplier: K_{dev}. Based on interviews with development experts at KE-works it was concluded that the ratio between actual coding and the complete time spent by a developer (K_{dev}) on all sub-activities could be viewed as constant. This is then multiplied with the hour rate of a developer, hence the estimated effort should thus be estimated for a developer with the skills and experience matching with the hour rate. Furthermore a similar Cost Matrix (CM_{dev}) as the one in Table 3.5 is applicable to the development cost with adjusted values for the development activities.
In this proposed methodology the estimation of development effort (E) is based on expert judgement of the knowledge engineer.

Integration cost
Many activities in the PDP are executed by means of an application (e.g. spreadsheet applications, Computer Aided Design (CAD) software). Upon the application of high levels of automation these applications require to be integrated in the process architecture. Hence, at specific levels of automation the cost of integration is required for the application to function automatically and for the WMS to communicate with the application and for example start it.
Based on previous projects of KE-works a classification is made of different integration classes. These classes can be seen in Table 3.6.

Table 3.6: Classification of integration types with accompanying costs

	Class 0	**Class 1**	**Class 2**	**Class 3**
Description	No integration needed.	Application has standard web API interface. (e.g. Optimus)	Application has a COM interface (e.g. Excel, CATIA)	No API available. Integration interface needs to be developed.
Integration cost (IC) [€]	0	4500	11000	35000

Besides the integration class also the experience with the specific application to integrate is of importance. If a similar application has been integrated before the cost of integration is expected to be lower. In equation 3.9 this effect is accounted for by the Learning Curve (LC). This value is 1 by default but could be adjusted by the knowledge engineer if a specific application is expected to have a different integration cost based on previous experiences.

$$C_{int_A} = IC_{class} \cdot LC \cdot (CM_{int_{ij*}} - CM_{int_{ij}}) \tag{3.9}$$

A WMS possibly supports integration with specific applications (i.e. the WMS has built-in adapters). In this case the integration cost is negligible. The same holds for the case in which the same application needs to be integrated for multiple activities. Therefore equation 3.9 should only be used for the calculation of integration cost for non-native applications and only one time per project per application.

Configuration cost
With an increasing level of automation, more knowledge is formalised and stored in a knowledge base. The

cost of formalising and storing the knowledge is already captured in the cost of knowledge acquisition. Frequently this knowledge base is not the system used by companies for storage of, for example, product models and process models. Hence, if a company increases the level of automation by, for example, developing handbook methods for the analysis, this knowledge is stored in a specific place. Furthermore if an application is developed, the knowledge used in this application also should be stored in a place other than in the direct code of the application itself. Therefore the configuration of the knowledge in the systems used by the company is required.
The effort required for this configuration again is dependent on the activity type and the current and proposed level of automation. The formula used in this research is presented in equation 3.10. The cost of configuration is closely related to the cost of knowledge acquisition and therefore the formula is similar. They differ in the coefficient K_{conf} and the coefficients of the Cost Matrix CM_{conf}.

$$C_{conf_A} = \bar{t} \cdot r_{KA} \cdot K_{conf} \cdot (CM_{conf_{ij^*}} - CM_{conf_{ij}}) \tag{3.10}$$

WMS license costs
This cost attribute is concerned with the license cost of the WMS, licenses of other applications are not taken into account. At lower levels of automation this cost is therefore not applicable. In this research it is assumed that the cost of maintenance and support is included in the license cost.
The cost per activity is determined using equation 3.11. The cost is based on the activity type (i), current level of automation (j), proposed level of automation (j^*) and cost of a license (C_{lic}).

$$C_{lic_A} = C_{lic} \cdot (CM_{lic_{ij^*}} - CM_{lic_{ij}}) \cdot [lic = 0] \tag{3.11}$$

The Cost Matrix (CM_{lic}) has a more binary nature for the cost of the license. The license is only applicable if the process is automated to such a level that a WMS is desired. In the case of the sample matrix in Table 3.7 the WMS license is only required for a level of automation 4.

Table 3.7: Sample license cost coefficients (CM_{lic})

	Level of Automation			
	1	**2**	**3**	**4**
Acquire	0%	0%	0%	100%
Process	0%	0%	0%	100%
Analyse	0%	0%	0%	100%
Decide	0%	0%	0%	100%
Implement	0%	0%	0%	100%

Furthermore the last term in 3.11, $[lic = 0]$, means that the license cost is only applicable if the the license has not been purchased before. If the knowledge engineer configures a process in which initially no license has been purchased, this license needs to be purchased for the first activity with a level of automation of 4 (using CM_{lic} of Table 3.7). For all subsequent activities the license cost is not added any more since the license has already been purchased. Resulting in one activity with license cost while possibly many other activities also use the license. In reality this cost of the license would be discounted over all tasks.
The cost of the license should be determined by the knowledge engineer and should be provided as an input variable. The license cost is based on multiple factors like the amount of users and the capabilities used by its customer.

Server costs
The third level of automation is defined in Table 3.2 as *"The activity is defined and the system suggests what to acquire and where it can be acquired. The source is responsible to acquire the items from the source. All information is available from a single source of truth."* For this single source of truth a server is required and therefore the accompanying costs need to be taken into account. To determine the server cost an equation similar to the equation for license costs is used and can be seen in Equation 3.12.

$$C_{serv_A} = C \cdot (CM_{serv_{ij^*}} - CM_{serv_{ij}}) \cdot [serv = 0 AND lic = 0] \tag{3.12}$$

Hence the knowledge engineer determines if a server is already available in the current process state. If this is true then no server costs will be made during the process. If it is false then, depending on the $CM_{serv_{ij}}$

the costs of a server is added. The term $[serv = 0 AND lic = 0]$ assures that no server cost is made if a WMS license or server already has been purchased. This is because the WMS license often includes single source of truth online storage.

Training costs
In reality it can not be expected that the users of a system can start working without any form of training (no matter how intuitive the developed tools or WMS). Therefore the cost of training is an important cost attribute. The training cost is, based on the current KE-works cost model, a fixed percentage of the sum of the other above mentioned costs. This percentage (K_{trg}) is determined by the Knowledge Engineer. The formula is presented in Equation 3.13 and the cost (C_{trg_P}) is determined for the full process (denoted with the subscript P).

$$C_{trg_P} = K_{trg} \cdot \left[\sum_{i \in \mathbf{z}} C_{KA} + C_{dev} + C_{int} + C_{conf} + C_{lic} + C_{serv} \right] \tag{3.13}$$

In Equation 3.13 the vector **z** is a vector of all activities in the process.

Management costs
The process of developing and implementing the automation initiatives requires management effort. The technical aspects needs to be managed to increase the likelihood of project success. But also management of the cultural aspects as explained in Section 2.5.5 is required. In this framework it is proposed that the management cost of the process can be modelled as a fixed percentage of the total project costs. This percentage needs to be determined by the Knowledge Engineer, based on experience and information on both technical and cultural challenges in the project. The equation used to determine the management cost of the innovation process (C_{mgt_P}) is shown in Equation 3.14.

$$C_{mgt_P} = K_{mgt} \cdot \left[\sum_{i \in \mathbf{z}} C_{KA} + C_{dev} + C_{int} + C_{conf} + C_{lic} + C_{serv} \right] \tag{3.14}$$

3.7.2. TOTAL AUTOMATION INVESTMENT COST

The total automation investment cost ($C_{inv_{total}}$)is the sum of all the cost attributes summed up for all activities in the process plus the cost of training and management. This is also displayed in equation 3.15

$$C_{inv_{total}} = \left[\sum_{i \in \mathbf{z}} C_{KA_A} + C_{dev_A} + C_{int_A} + C_{conf_A} + C_{lic_A} + C_{serv_A} \right] C_{mgt_P} + C_{trg_P} \tag{3.15}$$

In this research some cost attributes were purposely left out of scope. These cost include for example travel and stay costs when working out of the office. Upon using this framework the user should take them into account but they are hard to assess based on the process architecture.

3.8. INTEGRAL MODELLING APPROACH

By combining all elements as discussed in this chapter, a PDP can be modelled on an activity level with varying levels of automation. Therefore the following steps can be identified:

1. Analyse the PDP based on available documentation and interviews.
2. Construct a workflow on task level.
3. Determine the available resources and resource properties such as hour rate and availability.
4. Per task, determine the task lead time and define the activities performed in this task with their relative durations.
5. Per activity determine the current level of automation, required resources, development effort and optionally the integration application.
6. Assess the need for determining methodology matrices. If no matrices are available, or if matrices are not applicable to the process under investigation:

   Define the Duration Matrix for the process under investigation.

   Define the Cost Matrices for the process under investigation.

If all steps are successfully executed the PDP is modelled and can then be analysed. The analysis of the PDP based on the modelling approach as discussed in this chapter is the topic of the next chapter.

4 SIMULATION

The previous chapter discussed how to model the Product Development Process (PDP) and the methods used to analyse the performance of the PDP. This chapter explains how this model can be analysed for performance, taking into account PDP specific behaviour. In Section 4.1 the core concepts of Discrete Event Simulation (DES) are explained and why this simulation model is used. In Section 4.2, the structure of the simulation program is discussed. Finally, in Section 4.3, the behaviour of the simulation for specific PDP characteristics is discussed.

On a high level the simulation concept is illustrated in Figure 4.1. Here it can be seen that by means of the estimation methods, input parameters and the current levels of automation, by means of simulation, the process performance is analysed in terms of amongst others lead time, process cost, investment cost and other Key Performance Indicators (KPIs). Features are implemented in the simulator to account for important PDP characteristics, such as (number of) iterations, interruption, resource constraints and waiting time.

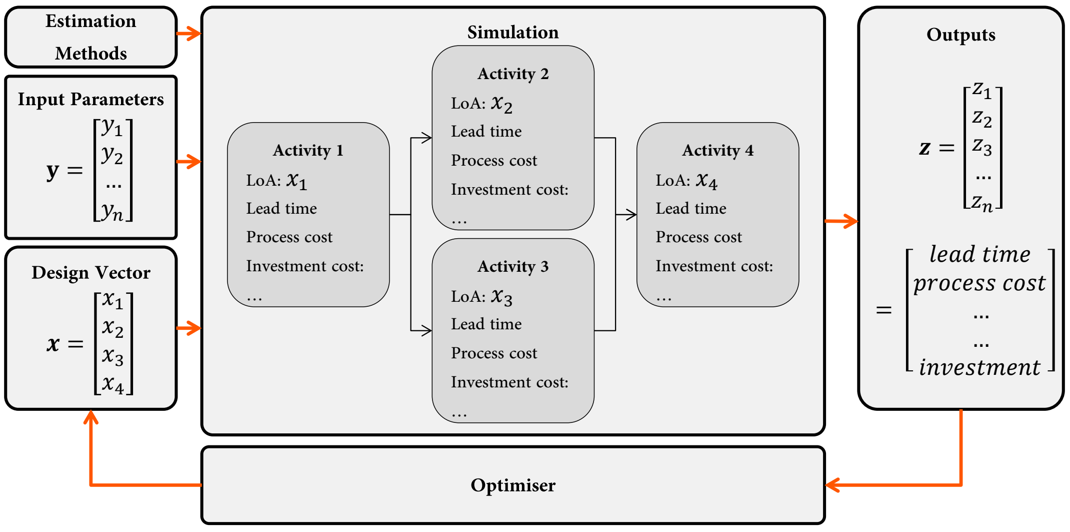

Estimation Methods
Input Parameters
$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_n \end{bmatrix}$
Design Vector
$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$
Simulation
Activity 1
LoA: x_1
Lead time
Process cost
Investment cost:
...
Activity 2
LoA: x_2
Lead time
Process cost
Investment cost:
...
Activity 3
LoA: x_3
Lead time
Process cost
Investment cost:
...
Activity 4
LoA: x_4
Lead time
Process cost
Investment cost:
...
Outputs
$\boldsymbol{z} = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ \dots \\ z_n \end{bmatrix} = \begin{bmatrix} \textit{lead time} \\ \textit{process cost} \\ \dots \\ \dots \\ \textit{investment} \end{bmatrix}$
Optimiser

Figure 4.1: High level overview of the simulation concept

4.1. DISCRETE EVENT SIMULATION

DES is a process-oriented simulation method, often used in modelling processes with discrete steps (e.g. warehouse inventory over time). In DES, state variables only change at specified points in time, referred to as events. The simulation jumps from event to event and skips the time when no events occur, hence no state variables are changed (similar to a mathematical step function). This is different from continuous simulation where the simulation makes small steps in time and adjusts for the changes occurring over this time. In this

research only discrete steps are of interest (i.e. only the starting and ending time), and therefore the DES is selected. An advantage of DES is the low computing power required, even in the case of processing multiple events in parallel.
The logic for the discrete event simulation engine is visualised in the pseudo code in Algorithm 1.

Algorithm 1 Pseudo code of discrete event simulation logic

```
Ensure: Ending Condition = false
Ensure: t = 0
 1: Initialise state variables
 2: Start all valid events
 3: while Ending Condition is false do
 4:     Set t to next event
 5:     Start next event
 6:     Update system variables
 7:     if All events processed AND (no events awaiting OR no possibility to process) then
 8:         Ending Condition is true
 9:     end if
10: end while
11: return All requested process information
```

During the simulation (i.e. in the while loop) the system variables are updated. This means that new events can be started or stopped for example.
In Algorithm 1 the Ending Condition is set to false initially and the simulation stops once this condition becomes true. The only reasons for the Ending Condition to become true is if either all events have been processed and no more events are awaiting or, if no events can ever be processed any more (e.g. a required resource for a process does not exist and hence the event will never meet its starting condition).

4.2. SIMULATION STRUCTURE

A similar algorithm as described above is used in the research at hand. This section will discuss the structure of the simulation as developed for this research. Note that this section only discusses the simulation itself and not the complete sequence (including constructing the workflow etc.) of the developed code, this full sequence is discussed in 4.5.
To allow a discussion of this simulation method a high-level Unified Modeling Language (UML) class diagram is presented in Figure 4.2. This is an indicative class diagram and does not fully represent the actual code structure but provides insight in the working principles of the simulation. The detailed diagram van not be shown due to confidentiality. In this section the two main elements of the simulation will be discussed, the environment and the activities.
The main element of this simulation is the *Environment*. This is the virtual environment in which the processes take place. Elements such as the resources, matrices for calculations, activities and containers are initiated in this environment. The elements need to be in the *Environment* to be able to communicate with each other, use each other and for the environment to be able to keep track of the event queue. All elements are initiated based on values provided by the user.
The *Environment* class has specific settings determining how the model should behave once a simulation is started. This includes, for example, settings on how to deal with iteration (as discussed in Section 4.3.3) or the hour rate of the knowledge engineer as defined by the user.
As can be seen in the class diagram the environment has function *run()*. This function starts the simulation process by stepping from event to event. At the start of the simulation the property *now* is equal to the starting time (default is zero). With every step this property is adjusted to match the artificial lead time.

The model uses a class (*Activity*) to model the activities in the workflow. These activities can be regular activities (i.e. implement, process, analyse, implement) or a gateway activities (i.e. decide), from now on referred to as an *Activity* instance and *Gateway* instance respectively. An *Activity* instance is modelled as a super-class and the *Gateway* instance inherits from this super-class. It can be seen in Figure 4.2 that the *Gateway* class has additional and adjusted properties and functions. The instances are subject to multiple constraints. Resource constraints and precedence constraints are the main constraints. At the start of the simulation (t=0,

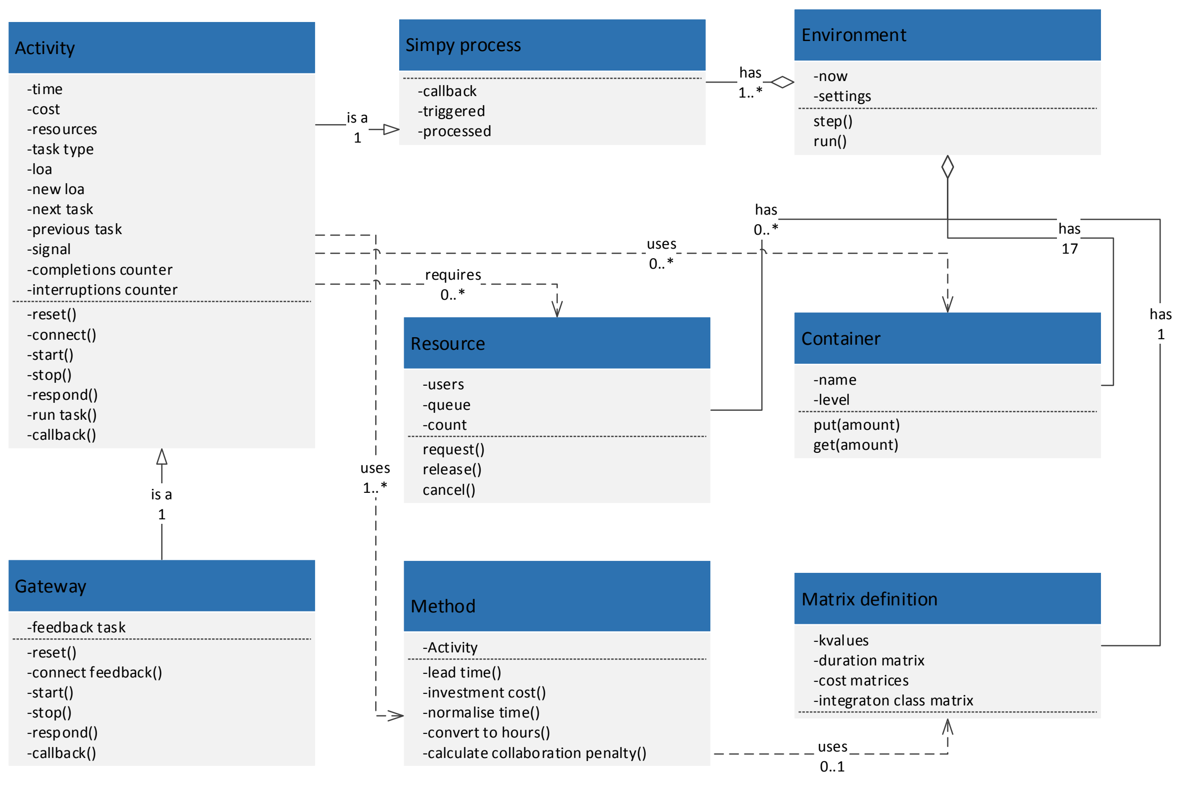


Figure 4.2: High-level UML class diagram of the developed simulation method

unless otherwise defined) all instances assess if all their constraints are met; the process of assessing this is referred to as *responding*. Once all constraints are met the instance starts and is completed after the duration. This duration is determined based on the inputs and by means of the method explained in Chapter 3. During the full duration the instance uses the required resources from the resource pool, hence no other instance can use the same resource at the same time. Upon completion it interacts with other *Activity* and *Gateway* instances by sending a signal to the succeeding instance, or instances, triggering them to respond.
The instances use methods to calculate the properties like lead time and investment cost. Due to the multitude of methods these have been placed in a separate utility toolbox in the program to enhance code structure (See *Method* class in Figure 4.2).

4.3. BEHAVIOUR

The developed DES algorithm distinguishes from other algorithms by the way it deals with complex PDP properties like iteration, rework and collaboration. These aspects are addressed in the following subsections.

4.3.1. ITERATION

During the initialisation, based on the inputs, the model determines whether an instance causes feedback. Determination is straight-forward: if an activity has an entry above the diagonal in the Design Structure Matrix (DSM), it provides feedback. If that is the case, the instance is of the class *Gateway*. The value of the entry above the diagonal is equal to the amount of iterations. The decision whether or not to feedback is made in the model based on the current state of the instance. The *Gateway* instance checks the total times it has been completed and verifies if that is below the set amount of iterations. If this is the case then the *Gateway* instance sends a *reset* signal to the predefined feedback instances, accompanied with a reference of the sender (i.e. the current *Gateway* instance). The instances receiving this *reset* signal stop their process if they are running. How the instances receiving the signal deal with this feedback is discussed in the next paragraph on rework modelling.

4.3.2. REWORK

Rework is modelled by sending reset signals between activities (i.e. *Activity* or *Gateway* instances). If an activity receives a *reset* signal, the activity is stopped and resets the progress of the activity to zero. Subsequently the activity forwards the signal to its succeeding activities to cause a trickle-down effect. This trickle-down effect accounts for the successive feed-forward rework as discussed by Cho and Eppinger [14]. This policy has been illustrated in Figure 4.3. As soon as the *Gateway* instance triggers the feedback, all the work performed by completed activities 1, 2, 3 and 4 is reset. Also the work in progress in activity 5 is stopped and reset to zero (i.e. the task is interrupted). It is important to note that it is assumed that rework in an activity always leads to rework in its succeeding tasks if the succeeding task has started or has already been completed before the reset signal. A *reset* signal will thus always trickle-down in the workflow. In reality this might not always be the case since rework of one task (e.g. increasing the bolt diameter) not necessarily leads to rework of a succeeding task (e.g. select bolt supplier).

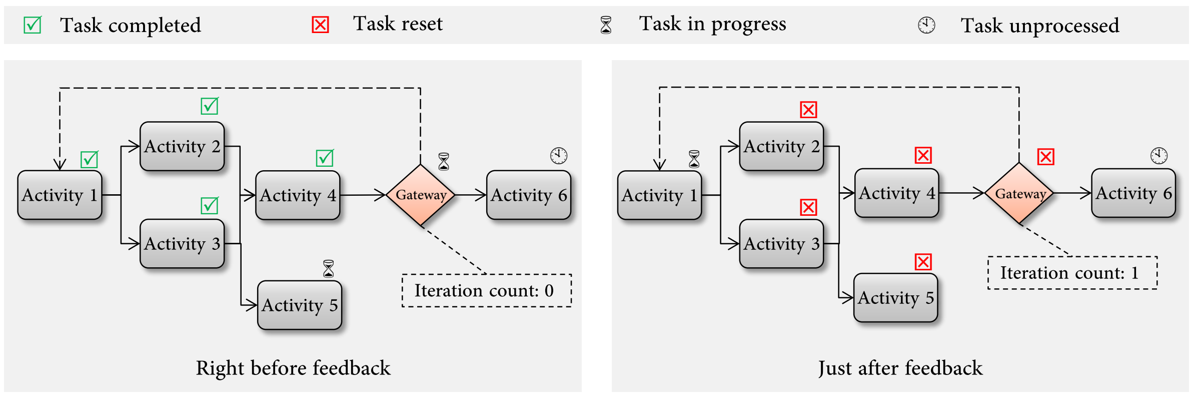


Figure 4.3: Workflow illustrating the rework modelling policy

In the case of a *Gateway* the *reset* signal is not unconditionally forwarded. The *Gateway* assesses the sender of the *reset* signal. If the signal is sent by itself the *reset* signal is not forwarded. If the *reset* signal is sent by another gateway the reset signal is forwarded and the internal counter of the gateway assessing the amount of iterations is then set back to zero.
This results in specific behaviour for processes with nested iterative loops. An example of such a process is provided in Figure 4.4. The behaviour of the example would yield that P_2 and P_3 are executed four times since the counter of the amount of iterations of P_3 is set to zero by the reset signal of P_4. The author acknowledges that this is not representative for every process but is the closest to reality taking into account the assumption on rework stated in Section 1.3.4.

	P_1	P_2	P_3	P_4
P_1	▨			1
P_2	1	▨	1	
P_3		1	▨	
P_4			1	▨

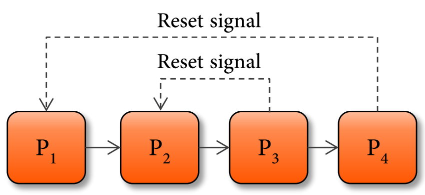


Figure 4.4: DSM and flowchart to illustrate the principle of rework and reset signals as used in the simulation

In computing the recurring cost (process cost), all time spent by resources on activities is taken into account. Hence the time spent on rework and interrupted activities all add to process cost.

4.3.3. COLLABORATION

Collaboration is modelled by using penalties for transactions between different resources. This penalty is a delay before starting the actual activity and is determined by the user as a standard penalty, constant over the full process. This constant, $t_{tostart}$, is set by the user in the setting input parameters as can be seen in Table G.2. The penalty is not applicable to every activity since such a delay is not realistic for every activity. Each activity verifies if the resources of the previous activities are a subset of its current resources, if this is not the case then the penalty is accounted for. This applies in the case human resources are involved. In case of fully automated activities, no delay is applied based on the assumption that the automated system has always a computer resource available.

This approach for modelling of collaboration by using penalties is strict. It could be the case that of 3 subsequent activities the second one is fully automated but has a negligible duration. This would result in a penalty for the third task. In reality the resource responsible for the first and last activity would "wait" the negligible time for the automated activity but no significant waiting time would be required. The strictness of this approach could be viewed upon as a downside of this way of modelling the collaboration, therefore in recommendations in Section 9.3 adjustments are suggested to reduce penalty strictness.

INFINITE ITERATIONS

By using the policies described above, it would be possible to configure a process with an infinite iterative loop. To cope with this behaviour the user is able to define the settings to be able to deal with this issue. Detection of this infinite loop is not possible before starting the simulation. Therefore the system assesses for each *Gateway* instance if the amount of completions of the instance does not exceed a given amount, defined by the user. If the amount of completions exceeds this value two options for model behaviour are provided to the user:

1. The simulation stops and displays an error.
2. The instance under consideration, a *Gateway* instance, is transformed to an *Activity* instance and hence is not able to provide feedback any more.

Another option is to select a third option. Here no infinite loop detection is used, the system behaves as such that infinite loops can never be configured.

3. If a *Gateway* instance receives a *reset* signal not provided by the itself (i.e. sender is not the current *Gateway* instance), then the amount of iterations of the instance is lowered by 1. Eventually resulting in 0 required iterations and hence the *Gateway* instance becomes a *Activity* instance.

4.3.4. RESOURCE ALLOCATION

In the PDP, and many other processes involving human resources, the number of resources is limited to a certain capacity. In most cases this is a monetary trade-off for a company. Due to this capacity it could happen during the process that the resource required for a specific activity is occupied at that moment in time. This would result in waiting time for the activity and could result in a longer lead time. Therefore it is important to take the limited amount of resources into account in the simulation process.

As can be seen in Figure 4.2, the *Resource* class has a property *count*, this is set at an initial value as defined by the user. Upon requesting, the count value is lowered by the amount of requested resources by a task. If the count is not at a sufficient value to provide the requested amount of resources for a process, the process needs to wait until another process releases its resources, hence inducing waiting time in the simulation.

4.4. SIMULATION OUTPUTS

By means of the described simulation multiple KPIs are computed for the configured process. These KPIs are calculated on an activity level and on process level (see Figure 3.1). The following indicators are calculated on a process level:

These indicators in Table 4.1 are stored in the *Containers* as indicated in Figure 4.2.

Furthermore a number of indicators are calculated on an activity level and are stored in the *Activity* instance itself[1]:

[1]Illustrated in Figure 4.2 by the properties *cost* and *time*

Table 4.1: Overview of computed KPIs on a process level

Parameter	**Description**
Total lead time	Total lead time from starting first activity to the finish of the last.
Total process time	Total amount of man hours invested during the process.
Time awaiting resources	Total time activities can not start due to resource constraints.
Time waiting to start	Total time added to the process due to collaboration penalty (see Section 4.3.3).
Total interrupted time	Total time activities have started but are interrupted before completing.
Cost of interrupted time	Cost of resources during total interrupted time.
Total process cost	Total cost of all resource utilisation during the process.
Number of projects until Break Even Point (BEP)	Number of project before the investment paid back itself.
Total investment cost	Total investment cost to achieve levels of automation of simulated process architecture.
- Total internal knowledge engineering cost	
- Total external knowledge engineering cost	
- Development cost	
- Integration cost	
- License cost	
- Server cost	
- Configuration cost	
- Management cost	
- Training cost	

Table 4.2: Overview of computed KPIs on an activity level

Parameter	**Description**
Activity lead time	Total lead time from starting the activity till it is finished for a single cycle (hence no iteration).
Activity process time	Total amount of man hours invested during the process, taking into account iteration and interruption.
Time awaiting resources	Total time the activity can not start due to resource constraints.
Time waiting to start	Total time added to the activity due to collaboration penalty (see Section 4.3.3).
Interrupted time	Total time an activity has started but was interrupted before completing.
Cost of interrupted time	Cost of resources during total interrupted time.
Activity process cost	Cost of the utilised resources by the activity.
Activity investment cost	Total investment cost to achieve the configured level of automation of the activity.
- Total internal knowledge engineering cost	
- Total external knowledge engineering cost	
- Development cost	
- Integration cost	
- License cost	
- Server cost	
- Configuration cost	
- Management cost	
- Training cost	
Amount of completions	Number of times the activity is completed.
Amount of interruptions	Number of times the activity is interrupted.

4.5. SIMULATION ACTIVITY SEQUENCE

In previous sections the simulation of a single simulation experiment is discussed (i.e. only one process architecture is analysed). Before starting this actual simulation of the process, it needs to be defined, imported and constructed in the SimPy environment. During this research a first prototype to assist the user in configuration, verification, simulation and interpreting the results is developed. This section is solely concerned with the activities inside the Python environment. The sequence of these steps in the program are displayed in Figure 4.5. The first header is the user of the system, starting the simulation. The remaining four headers are classes in the prototype. The last header, *Simulation environment*, refers to the *Environment* in Figure 4.2.

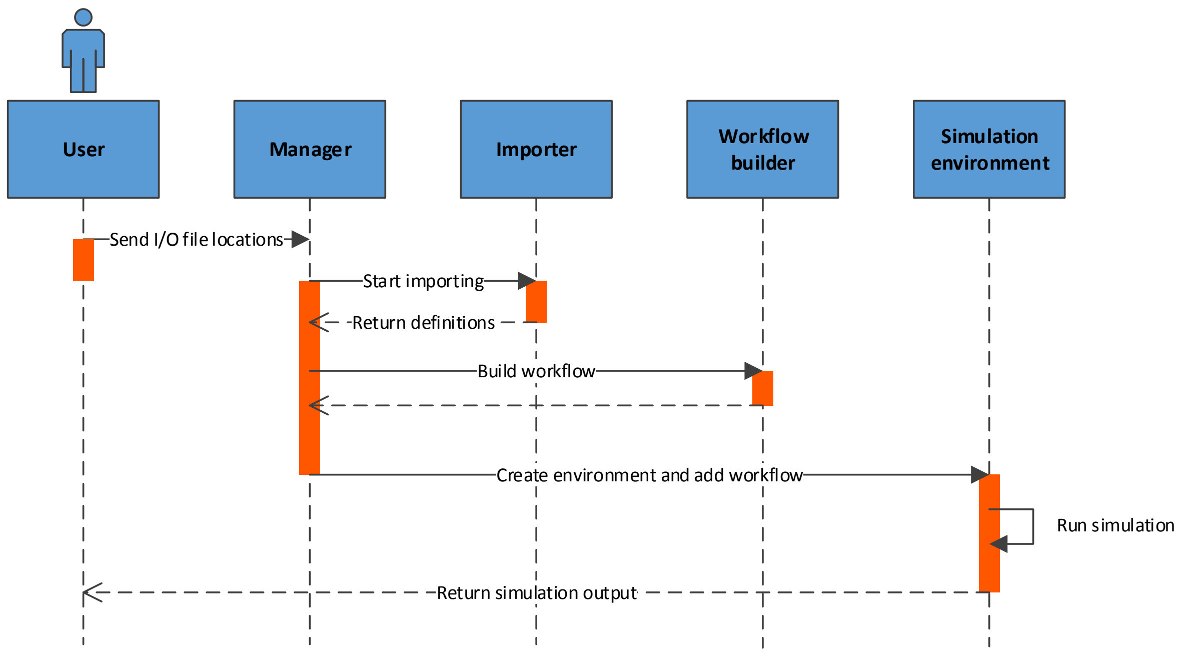


Figure 4.5: UML sequence diagram for the developed simulator

It can be seen in Figure 4.5 that the user first sends the *Manager* information on the input and output file locations. The *Manager* subsequently uses the *Importer* to interpret the inputs and retrieves the process definitions. Based on these process definitions the *Workflow builder* starts to build a workflow, verifies if all inputs are valid and returns it to the *Manager*. This workflow is provided to the *Simulation environment* and performs the simulation as discussed in this chapter.

As stated in Section 1.3.1, it is the objective to develop an optimisation method. The simulator has the ability to perform an exhaustive search (discussed in more detail in Chapter 5). This is indicated in the activity diagram for the developed simulator, as can be seen in Figure 4.6. Furthermore the verification of provided input is performed right after creating the environment. In further development of this code one could consider replacing it to right before the decision on whether or not to perform an exhaustive search.

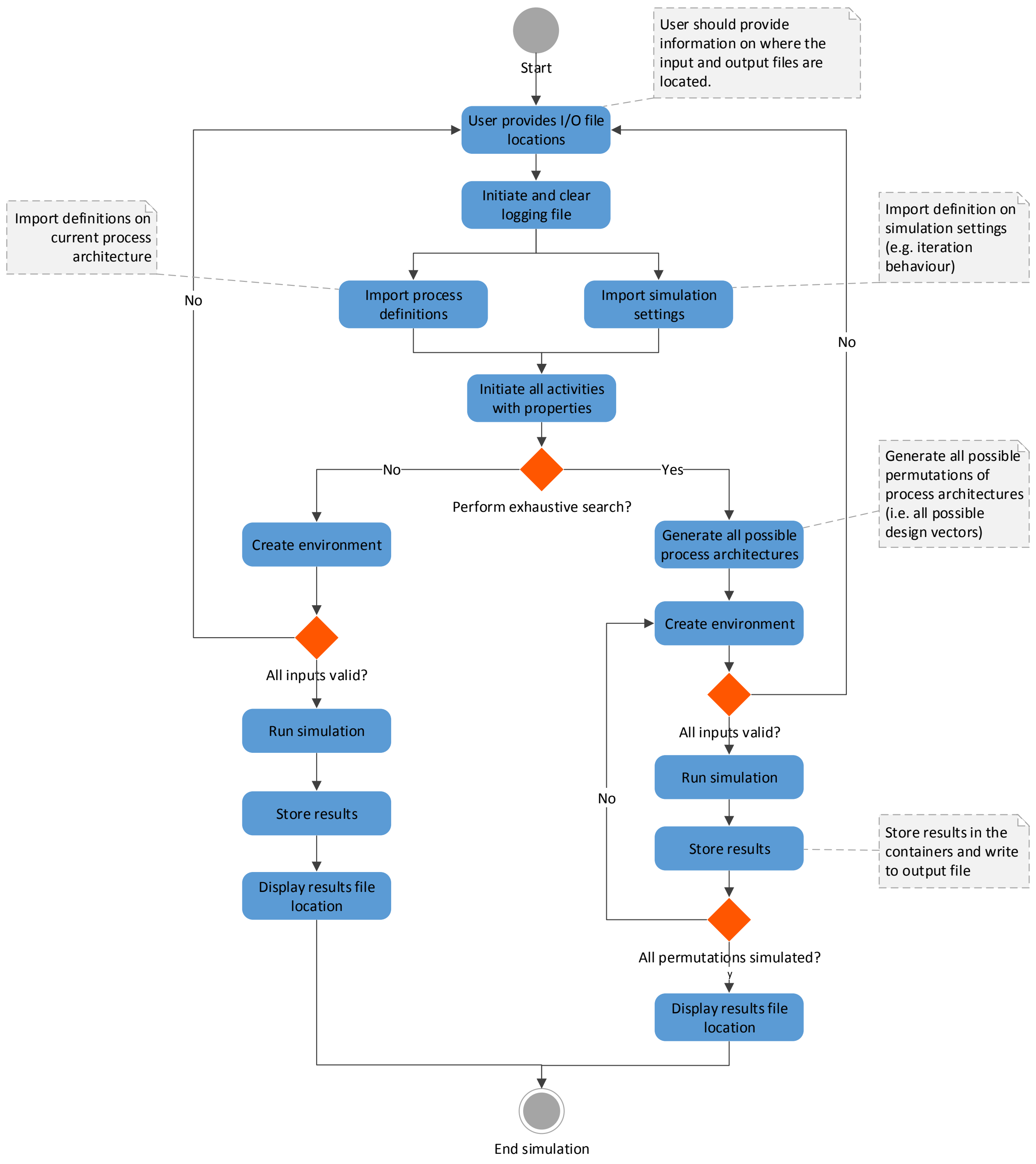

Start
User should provide information on where the input and output files are located.
User provides I/O file locations
Initiate and clear logging file
Import definitions on current process architecture
Import definition on simulation settings (e.g. iteration behaviour)
Import process definitions
Import simulation settings
No
No
Initiate all activities with properties
No
Yes
Perform exhaustive search?
Create environment
Generate all possible process architectures
Generate all possible permutations of process architectures (i.e. all possible design vectors)
Create environment
All inputs valid?
All inputs valid?
Run simulation
Run simulation
Store results
No
Store results
Store results in the containers and write to output file
Display results file location
All permutations simulated?
Display results file location
End simulation

Figure 4.6: UML activity diagram for the PDP simulator

5
Optimisation strategy

The simulator described in the previous chapter is able to analyse any given process architecture and output important process Key Performance Indicators (KPIs) like lead time and investment cost. The goal of this research is to provide insight in the trade-off between costs and benefits for the use of different levels of automation. The trade-off of these costs and benefits results in a Multi-Objective Optimisation (MOO). The optimisation of these multiple objectives is the main topic of this chapter.
First the optimisation problem is explained in Section 5.1. In Section 5.2, the concept of MOO is discussed and defined more elaborately. Subsequently it is discussed why an optimiser is needed in Section 5.3. Following this, in Section 5.4, the requirements for a MOO algorithm for the problem at hand are discussed. The selected algorithm is then briefly introduced in Section 5.5.

5.1. Optimisation problem

In making a trade-off between costs and benefits multiple objectives can be identified. Some users would like to reduce the lead time, no matter what while others might want to reduce process cost but only until a specific investment cost. Since no information is present on the relative importance of the different objectives it is not possible to determine one single solution to be the optimal solution for the problem at hand. A set of optimal solutions is determined, also referred to as the Pareto front. These Pareto optimal solutions are the process architectures located on the Pareto front as indicated in Figure 5.1. Each process architecture is unique and varies in the level of automation of the activities, resulting in an optimum value of the objective functions at hand. Two example process architectures can be seen in Figure 5.2.

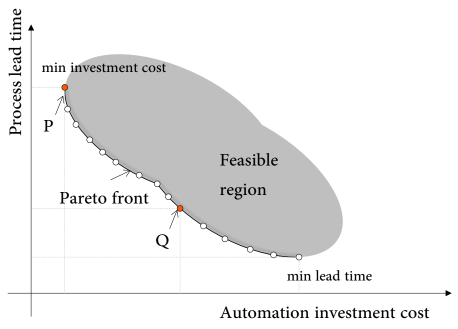


Figure 5.1: Pareto front trading off lead time and investment cost

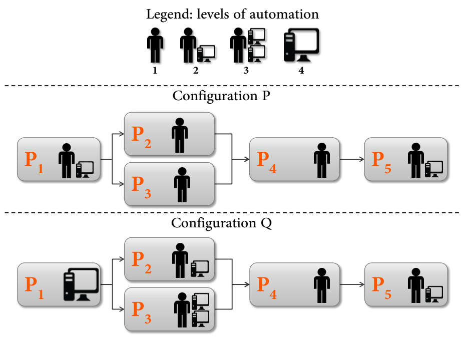


Figure 5.2: Simplified illustration of the process architectures P and Q on the Pareto front

It can be seen in Figure 5.2 that the process structure is identical and solely the levels of automation on an activity level are adjusted. This illustrates the optimisation problem in this research: defining the process architectures (i.e. level of automation of each activity) such that they result in a Pareto optimal solution. This Pareto front can then be used in trading-off different process architectures.
In the example above the objectives lead time and investment cost are selected but any combination of KPIs

listed in Table 4.1 can be used. The optimisation of the combination of these KPIs is defined as a MOO and is discussed in the following section.

5.2. MULTI-OBJECTIVE OPTIMISATION (MOO)

In MOO more than one objective function needs to be optimised simultaneously. These objective functions might conflict with each other, as is the case in the MOO in this research. As discussed in Section 2.2 cost, time and quality compete with one another. A higher level of automation generally leads to a lower lead time but also to higher investment cost.
A Multi-Objective Optimisation problem is mathematically defined as in equation 5.1.

$$\begin{aligned} &\underset{x\in\Re^n}{\text{minimize}} \overrightarrow{f}(x) = (f_1(x),\ldots,f_k(x))^t \\ \\ &\text{subject to:} g(x) \geq 0 \qquad\qquad where: g:\Re^n \to \Re \\ &\qquad h(x) = 0 \qquad\qquad h:\Re^n \to \Re \\ &\qquad x_l \leq x \leq x_u \qquad\qquad x_l, x_u \in \Re^n \end{aligned} \tag{5.1}$$

Here x is the design vector, in the case of this research the levels of automation of the various Product Development Process (PDP) activities. $\overrightarrow{f}(x)$ is the vector containing multiple objective functions to be minimised. This optimisation is subject to specific inequality and equality constraints and lower and upper limits of the design vector. In this research the design vector is limited for each variable (i.e. activity level of automation) by a lower bound on the level of automation as the current level of automation. The upper bound is constrained by either the maximum level of automation or a level of automation as provided by the user.
The goal of the MOO is to find the design vectors leading to Pareto optimal solutions taking into account the constraints and boundaries on the design vector. An example of a Pareto front can be seen in Figure 1.3, the Pareto front is in this case represented by a solid continuous line. In this research the Pareto front will not be a continuous line due to the discrete nature of the variables in the design vector. A design vector is called Pareto optimal if no other $\overrightarrow{x} \in \Re^n$ such that $f_i(\overrightarrow{x}) \leq f_i(\overrightarrow{x}^*)$ for all $i = 1,\ldots,k$ and $f_j(\overrightarrow{x}) \leq f_j(\overrightarrow{x}^*)$ for at least one j.

5.3. NEED FOR OPTIMISATION

In order to generate this Pareto front, a possibility is to perform an exhaustive search (brute-force search); this would imply assessing all possible permutations of the design vector, hence all the possible combinations of levels of automation for each activity of the process. The total amount of possible permutation experiences a combinatorial explosion with an increasing number of activities. To give an indication an exhaustive search for 20 activities with 4 levels of automation to assess for each activity would result in 4^{20} experiments (1099511627776). Even with a simulation times per experiment of 1 millisecond this would still result in over 17 years of computational time using a single computer! Therefore it is clear that exhaustive search would result in too long computational time for the proof of concept purpose of this framework; hence a MOO search algorithm is preferred.

5.4. ALGORITHM REQUIREMENTS

The MOO search algorithm needs to be able to assess different design vectors and intelligently search in the right direction for the process architectures resulting in optimal architectures (i.e. Pareto optimal solutions). This intelligent searching algorithm should take into account the constraints and assess aspects like the convergence criteria after each evaluation. Due to the discrete modelling of the levels of automation a gradient-based search method can not be used. Evolutionary algorithms are a type of meta heuristic search method able to escape from a local optimum. These algorithms are deemed suitable to solve the MOO because they can deal with multiple solutions per iteration. Therefore per iterations multiple Pareto optimal solutions can be found whereas traditional mathematical optimisation methods would require multiple runs. Furthermore the evolutionary algorithms are less sensitive to the shape of the Pareto front and are able to deal with discontinuous and concave shapes for example. Therefore an evolutionary algorithm is suggested. Many different algorithms are available, in this research a Non-dominated Sorting Genetic Algorithm is selected which is discussed in the following section.

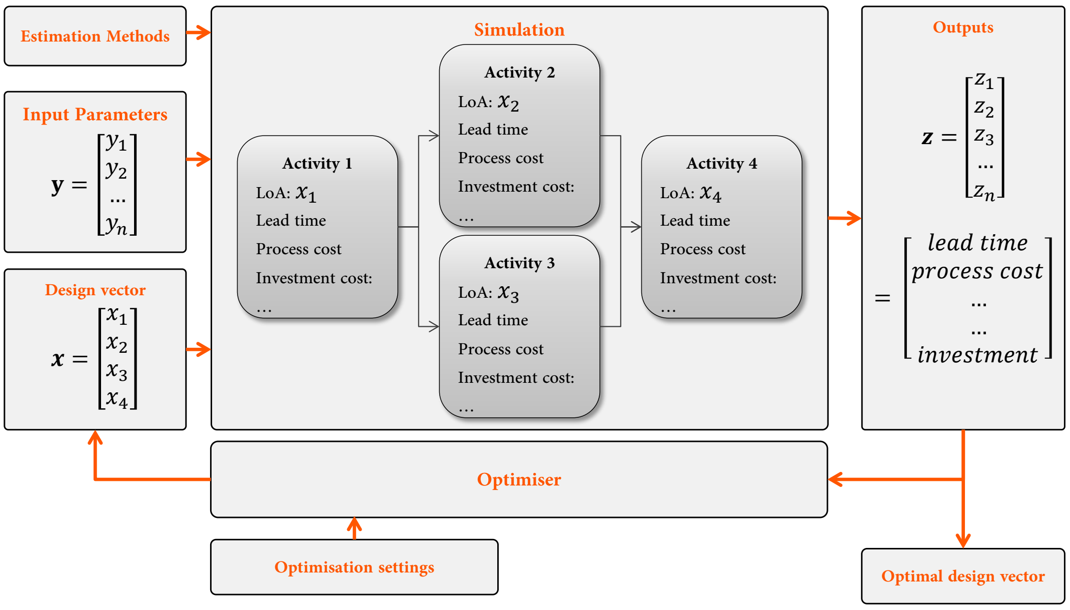


Figure 5.3: High level example of the optimisation system

In Figure 5.3 the implementation of an optimiser using a algorithm is illustrated on a high level. The optimiser is coupled to the simulator and hence the optimiser is able to adjust the design vector and to interpret the outputs. Based on the output the Genetic Algorithm (GA) determines the design vector for the coming iteration. This GA is discussed in the following section.

5.5. NON-DOMINATED SORTING DIFFERENTIAL EVOLUTION ALGORITHM

The algorithm used in this research is based on the NSGA-II algorithm developed by [4]. The Non-dominated Sorting Genetic Algorithm (NSGA) as developed by [51] received some criticism due to the high computational complexity, the lack of elitism and the need for specifying the sharing parameter. Therefore [4] proposed an improved version of the NSGA called NSGA-II.

This improved algorithm differentiates itself from the predecessor with three main aspects. Firstly it uses a fast non-dominated sorting procedure, secondly an elitist-preserving approach and thirdly by a modified main loop. The first two aspects will be discussed briefly in the following paragraph. A more detailed discussion the algorithm can be found in Appendix C.

A fast sorting algorithm is implemented to be able to sort all experiments and assess their domination. With a large number of experiments this algorithm reduces computational complexity. The elitist preserving approach is used to give priority to the reuse (in the genetic algorithm) of experiments in less crowded regions (i.e. with few other experiments in proximity). This increases the spread of experiments throughout the design space.

6
Integrated simulation and optimisation framework

Previous chapters discussed all individual elements used in the proposed methodology. In this chapter the integration of the elements in a framework is discussed. It is discussed how the elements interact with each other and how the user interacts with the framework.
The elements discussed in previous chapters have been implemented in the integrated framework illustrated in Figure 6.1. The integrated framework assists in the knowledge acquisition, structuring, simulation and optimisation of the process architecture. It can be seen in Figure 6.1 that multiple software applications are used in this framework. In the following paragraphs the elements are discussed.
Section 6.1 discusses the part of the framework in which the process and simulation are configured. Section 6.2 discusses the implementation of the simulation in the framework. Next, in Section 6.3, the optimisation is discussed. Finally, in Section 6.4, the performance of the integrated framework is reviewed.

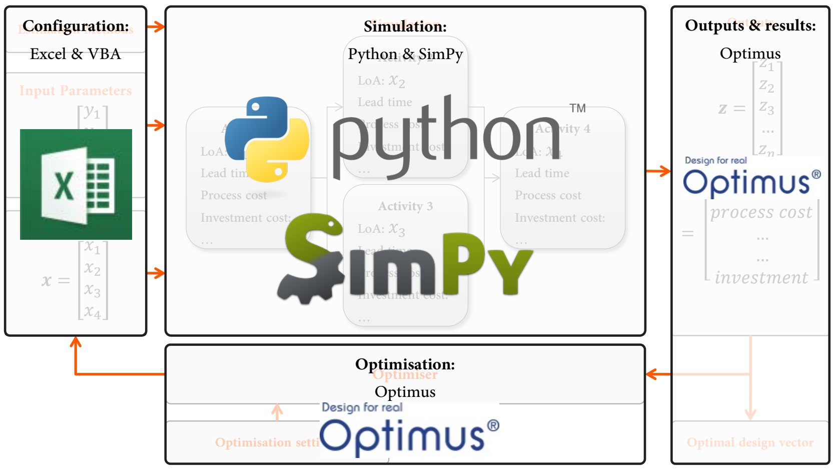


Figure 6.1: Overview of the integrated simulation and optimisation framework

6.1. Definition and configuration platform

In the first step the Product Development Process (PDP) is defined and its specific settings are stated. To this purpose, Microsoft Excel in combination with custom Visual Basic for Applications (VBA) scripts is used to provide a user friendly, and partly automated, interface to define the process to be analysed and optimized. The Excel file provides a structured method to capture all the process information. It is developed as such that it provides visual feedback on invalid or missing information. VBA scripts are used to automatically transform the high-level task overview (i.e. list of tasks with relative time distributions of activities) to a list of all activities in the process. The scripts also automatically create a Design Structure Matrix (DSM) based on task order in the Excel file, here the user is solely responsible for the definition of task precedence constraints.

The Excel interface also allows the user to directly run the simulation for the process as defined in the sheets. Using this method the user does not need to interact with the simulator (code) itself. Once the run is completed the interface prompts information on the process and the location of more detailed process analysis.

6.2. Simulation and analysis platform

The PDP simulator is developed using the open-source language Python and uses a Discrete Event Simulation (DES) library named SimPy [52]. SimPy is an open-source library for process-oriented DES; it is written in Python and also being called from Python. SimPy exploits generators (i.e. specific type of Python iterator) to yield specific events in time. This specific library is chosen due to the high adjustability and processing speed. This adjustability allows to connects with frameworks of partners involved in this research.
The simulator is provided with process information and settings by the Excel file. The file contains all information regarding the process but also settings for the simulator itself. Therefore no settings are hardcoded in the simulator itself to improve maintainability and clarity for current and future users of the tools.
In the settings the user can determine to perform one experiment or to perform an exhaustive search. This provides the user with the option to also generate a Pareto front without the use of an external optimiser. But as discussed in the Section 5.3 an exhaustive search quickly yields a high computational time and therefore the use of an optimiser is advised.
The simulator stores the results in project dictionaries. To improve usability of the information, the results are also exported to a .txt file and and .xls file. The .xls file is mainly of added value in the case of an exhaustive search without the use of an external optimisation platfrom (e.g. Optimus).
If no optimisation platform is available the simulator can still be used to perform single experiments. This is of use in analysing a specific process architecture without the need of optimisation.

6.3. Optimus platform

For the optimisation the Optimus platform developed by Noesis Solutions is used. This platform allows for the integration of multiple software tools such as Microsoft Excel and Python in this case. As can be seen in Figure 5.3 the optimiser interacts with the inputs and outputs of the simulator. The Optimus platform is able to interact with the configuration platform and adjust the design vector and is able to retrieve and interpret the results of the simulator. In the Optimus platform a workflow is generated for each specific process. An example workflow can be seen in Figure 6.2. The elements of this workflow are identical for each optimisation in this research but the content differs per process.

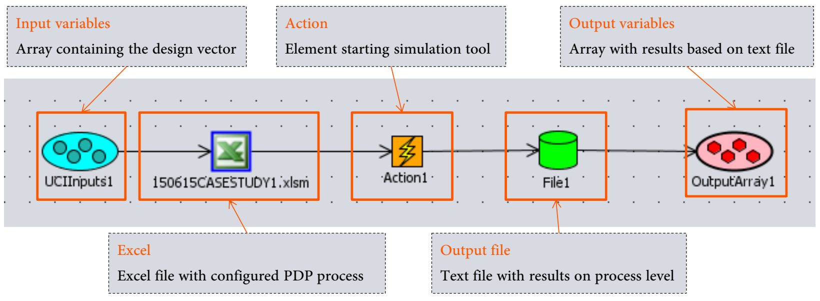
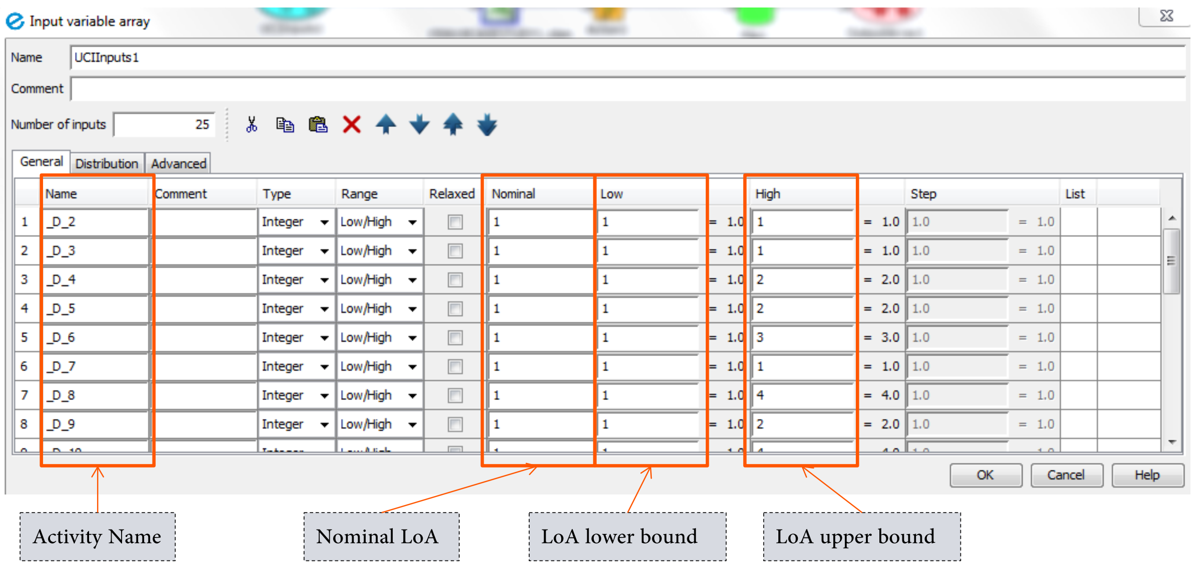

Input variables
Array containing the design vector
Action
Element starting simulation tool
Output variables
Array with results based on text file
UCIInputs1
150615CASESTUDY1.xlsm
Action1
File1
OutputArray1
Excel
Excel file with configured PDP process
Output file
Text file with results on process level

Figure 6.2: Workflow as displayed in the Optimus GUI

Within this workflow, five elements can be identified. An input variables element, an Excel element, an action element, an output file element and an output array element respectively from left to right. In the input variables element the design variables of the design vector are listed. For each item in the design vector information is provided on for example boundaries. The interface in which this information is displayed and possibly adjusted can be seen in figure 6.3. It can be seen that the type of variable (integer or real) and boundaries are displayed.

The elements in the design vector have a mapping on the second element, the Excel element. The platform adjusts the actual values in the cells in the Excel file that correspond with the design vector. For every experiment the Excel file with unique design vector values is provided to the action block which runs the Python application with a specific design vector and other parameters as provided in the Excel file. The outcomes



Name: UCIInputs1

Comment:

Number of inputs: 25

	Name	Comment	Type	Range	Relaxed	Nominal	Low		High		Step		List
1	_D_2		Integer	Low/High	☐	1	1	= 1.0	1	= 1.0	1.0	= 1.0	
2	_D_3		Integer	Low/High	☐	1	1	= 1.0	1	= 1.0	1.0	= 1.0	
3	_D_4		Integer	Low/High	☐	1	1	= 1.0	2	= 2.0	1.0	= 1.0	
4	_D_5		Integer	Low/High	☐	1	1	= 1.0	2	= 2.0	1.0	= 1.0	
5	_D_6		Integer	Low/High	☐	1	1	= 1.0	3	= 3.0	1.0	= 1.0	
6	_D_7		Integer	Low/High	☐	1	1	= 1.0	1	= 1.0	1.0	= 1.0	
7	_D_8		Integer	Low/High	☐	1	1	= 1.0	4	= 4.0	1.0	= 1.0	
8	_D_9		Integer	Low/High	☐	1	1	= 1.0	2	= 2.0	1.0	= 1.0	

Figure 6.3: Overview of input parameters as displayed in the Optimus GUI

are then exported to the fourth element, the output file element. A specific mapping is defined to be able to identify the output variables and to map them on the right elements in the fifth element (the output array element). With this mapping the optimiser is able to assess the value of the different objective functions as defined by the user and is able to use this information in a subsequent iteration.
The optimisation algorithm as discussed in Section 5.5 is able to handle more than two objective functions. Using the Optimus platform it is thus possible to also perform a Multi-Objective Optimisation (MOO) using for example lead time, investment cost and other objective functions. It should be noted that computational times increase significantly with the number of objective functions, therefore the MOO is limited to two objective functions in this research.

6.4. INTEGRATED FRAMEWORK PERFORMANCE

It should be noted that the integration between the elements has an negative influence on the framework computational performance. The interfaces between the three elements lead to an increase in computational time. For every experiment the framework constructs the simulation environment and therefore extracts this information from the Excel file. The (re)construction of the process and environment has a significant effect on experiment lead time. Since the Optimus interface can only be used in combination with the non-exhaustive search setting in the simulator this (re)construction is required every experiment. This results in an experiment lead time of roughly 2 seconds for the integrated framework[1] whereas the experiment lead time is close to 0,1 second for the exhaustive search solely using the simulator[2]. This difference can be explained due to the absence of an optimisation algorithm in the exhaustive search and only a single communication with the Excel interface.
Hence room for improvement exists to improve the computational performance of the integrated framework. Therefore the simulator should be adjusted to be able to store the extracted information from Excel in the Python environment and hence eliminate the need to interface with Excel for each experiment.

[1] For an optimisation with 500 experiments
[2] For an exhaustive search with 70000 experiments

7
Framework verification and validation

This chapter discusses the verification and validation of the simulation and optimisation framework proposed in the previous chapters. Verification is the process of verifying that the developed solution meets the requirements and specifications. Hence this step is concerned with assessing the performance of the outcome of the framework with the specifications for this framework. It checks whether the inputs correspond with the outputs based on the used methods. Validation is concerned with checking if the results match with the known (reference) values. For example by comparing results with experiments or mathematical formulations.
First in Section 7.1 the simulation itself is verified by means unit testing the developed code. Subsequently in Section 7.2 the optimisation itself is verified by means of multiple test cases. Next in Section 7.3 the reliability and consistency of the optimisation is treated. Finally in Section 7.4 the validation of the framework is discussed. Due to scarce validation data this section is of a qualitative nature and is more a discussion on how to validate then a real validation of the framework based on reference data.

7.1. Simulation verification

In this section the results of the analyser are verified. It is of vital importance that each simulation experiment provides correct results as expected based on the theory discussed in Chapter 3.
The simulation algorithm can handle many different process architectures. Factors of influence are the constraints of precedence, resource availability, iterative behaviour, collaborative behaviour and the settings of the process as determined by the user. The combination of these factors leads to a large amount of test cases of which a selection is discussed in this section.
The simulation is verified by means of unit tests. This means that all blocks of code are independently tested on a function or method level. Hence all methods used to calculate costs or (lead)time durations are verified on a functional level.
Besides verification on a function level also the behaviour of the simulation as a whole has been verified. This is done by means of simulation of specific process architectures which consist of a comprehensible amount of activities (i.e. below 10 activities) of which the expected outcome under varying inputs can be mathematically calculated. These specific process architectures are determined as such that the set of these processes cover the expected architectures during real simulation. An overview of the code coverage report of the unit tests is provided in Table 7.1. This table shows that using the tests on average over all statements, 92% of the code is evaluated by testing without errors. .
The statements remaining uncovered (8 %) during the unit tests are merely conditional statements with the purpose of returning error messages. These are not encountered during the tests since the tests are all performed without errors.

7.2. Optimisation verification

Previous section verified the separate experiments by using the developed simulation algorithm. This section will use this verified simulation algorithm in the integrated framework including optimisation. A number of

Table 7.1: Code coverage report for unit testing of developed analyser

Module	statements	missing	excluded	coverage
Create Environment	48	4	0	92%
DSM Interaction	108	7	0	94%
Gateway Class	57	6	0	89%
Importer	10	0	0	100%
Input validation	67	27	0	60%
Logging	15	0	0	100%
Methods	134	4	0	97%
Task Class	180	26	0	86%
Tests	388	6	0	98%
Create Workflow	31	0	0	100%
	1038	80	0	92%

simple test cases are discussed here to illustrate the functionality of the framework. A much more complex test case from an industrial application is discussed later in Chapter 8.
The following test cases are all based on the same process configuration, referred to as base case. In each test case, one characteristic (e.g. resource cost, amount of iterations) is adjusted to demonstrate the difference in behaviour of the system. Flowcharts of the test cases are illustrated in Figure 7.1. All tasks have a duration of 20 hours and consist of all activity types (i.e. task 1 can be split up in 16,6% acquire, 16,6% pre-process etc.). Hence the total amount of activities equals 24 (4x6).

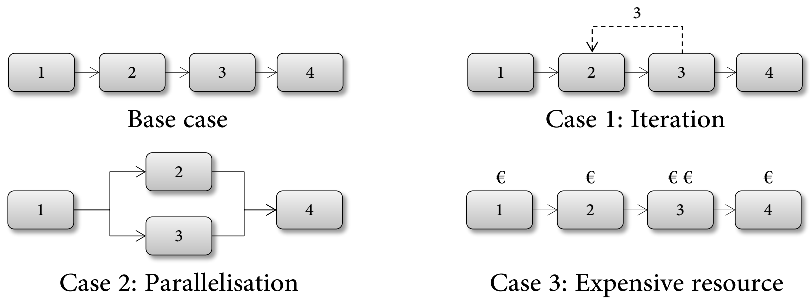


Figure 7.1: Flowcharts of the test cases for optimisation verification

7.2.1. Test case 1: Iteration

This case displays the effect of iteration in a process on the optimisation outcome. One iterative loop, of two iterations, is added in which activity 3 feedbacks to activity 2. In Figure 7.2 it can be seen that the Pareto front has shifted. Due to the iteration the process has a longer lead time. The slope of the Pareto front also has changed, implying that a larger lead time reduction than the base case can be obtained for a certain investment in automation solutions. This slope eventually matches with the slope of the base case when the iterative tasks have been fully automated.

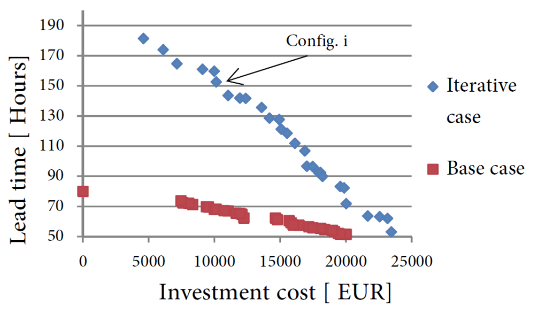


Figure 7.2: Pareto front for the lead time and investment cost for the base case and iterative case

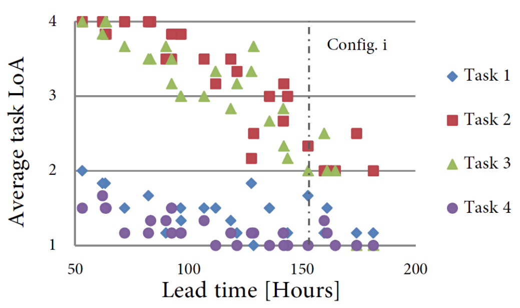


Figure 7.3: Average level of automation per task for the iterative case

Upon inspection of the levels of automation on the various activities in Figure 7.3 it can be seen that the points on the Pareto front correspond to process architectures with increasing levels of automation of the tasks involved in the iteration. These results match the expectations of the interviewed experts. In Figure 7.3 the points on 'vertical lines' correspond to a process architecture in Figure 7.2.

7.2.2. TEST CASE 2: PARALLELISATION

In this case the precedence constraints on activities are changed and activities 2 and 3 can be processed in parallel. This case is investigated for two different scenarios since resource availability influences the results. In scenario I only one resource is available, scenario II has two resources. In Figure 7.4 it can be seen that the Pareto front of scenario I is similar to the base case. This is as expected since with only one resource the process is not able to process parallel activities. For scenario II, however, a different front can be seen. The Pareto front flexes at a lead time of 30 hours. Upon investigation of the automation initiatives this can be explained. In Figure 7.5 the average levels of automation at different points on the Pareto of scenario II are plotted. Here it can be seen that at longer lead times (i.e. on the lower right side of the Pareto front) the non-parallel activities are first automated. Once these non-parallel activities have been (fully) automated the parallel activities (2 and 3) subsequently increases the level of automation. This is in accordance with the expectations since a higher level of automation for a parallel activity only becomes effective for a lower lead time if the other parallel task is also automated.

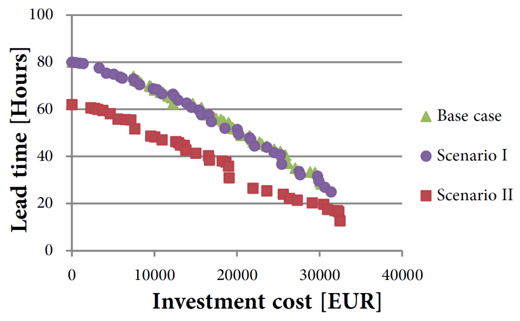


Figure 7.4: Pareto front for the lead time and investment cost for the base case and parallelisation case

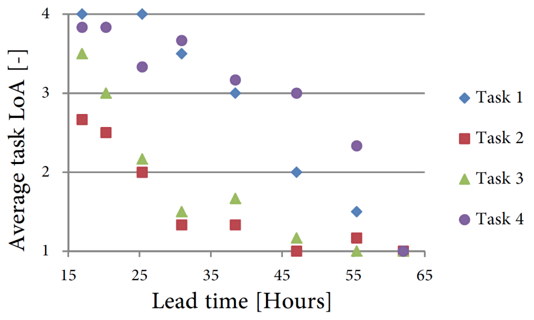


Figure 7.5: Selection of architectures on the Pareto front for Scenario II with average levels of automation per task

7.2.3. TEST CASE 3: RESOURCE COST

In this case one task utilises a different resource with a higher cost (in this fictitious case 100%). In this case it would be of no use to perform the Multi-Objective Optimisation (MOO) for lead time and automation investment cost since resource cost has no effect on lead time and only a relative small effect on automation investment cost. Therefore a MOO for the lead time and the number of projects needed until Break Even Point was performed.

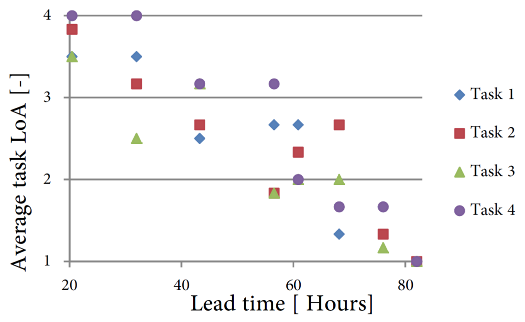


Figure 7.6: Average level of automation per task for the base case

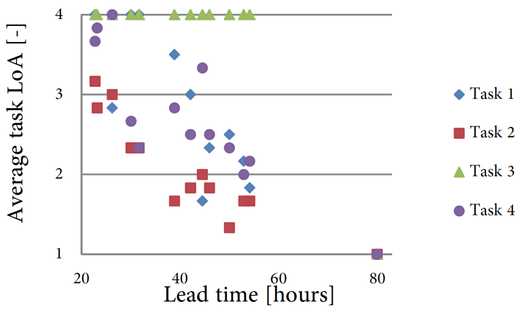


Figure 7.7: Average level of automation per task for test case 3

Figure 7.6 displays a selection of architectures on the Pareto front where it can be seen that all tasks are incrementally automated. When the cost of the resource of Task 3 is increased, a different graph is generated. This is illustrated in Figure 7.7, where it can be seen that Task 3 (using the expensive resource) is always fully

Table 7.2: Overview of the optimisation parameters applicable to the proposed optimisation method

Optimisation parameter	**Description**
Target front size	Total desired number of points on the Pareto front.
Population size	Number of experiments per iteration.
Start population	Design vectors used in the first population.
First population size	Number of experiments in the first iteration.
Random seed	Number used to initialize a pseudorandom number generator.
Weighting factor	Weighted differences of the variables of randomly selected designs from the previous generation. Low value leads to longer process but with a higher probability of global convergence.
Inverse crossover probability	Probability that genes are taken from predecessors without adjustments. Low values increase the risk of getting trapped in a local optimum.
Maximum number of iterations	Number of iterations before solution needs to be converged.

automated. Hence, as expected, the optimization framework suggests automating first the task with high resource cost, ceteris paribus.

7.3. OPTIMISATION RELIABILITY

In this research the optimisation method described in Chapter 5 is used. The method uses NSGA-II which includes tournament selection, recombination and mutations. These elements contain stochastic variables and therefore the Pareto front as generated might differ based on the optimisation settings but also under the same settings the Pareto front might differ due to these probabilistic elements. Reliability is the extent to which a measurement instrument yields consistent, stable, and uniform results over repeated observation or measurements under the same conditions each time [53]. This section will discuss the reliability of the optimisation. It does not discuss the correctness of the generated solutions but solely the reliability of these results. This will be discussed using an extended version of the base case as described above. This extended case is illustrated on a high level in Figure 7.8 and detailed process configurations can be found in Appendix D.

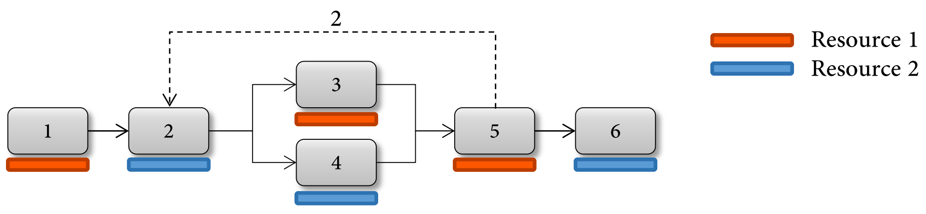


Figure 7.8: Flowchart of the process used for verification

Using this case the sensitivity of the optimisation settings are discussed in Section 7.3.2 and subsequently robustness is discussed in Section 7.3.3. The elements to vary and influence the optimisation (i.e. the optimisation settings) are first discussed in Section 7.3.1.

7.3.1. OPTIMISATION SETTINGS

In the optimisation method used in this research several settings are available to adjust the optimisation. The settings available in the optimisation platform as discussed in Section 6.3 are listed in Table 7.2.
Some of these settings will be adjusted in the following paragraph to assess the influence of these parameters on the optimisation process and outcome (i.e. the Pareto front). In this analysis of optimisation reliability the maximum total amount of experiments (i.e. the product of the maximum number of iterations and

population size) is limited to 2500 due to computational complexity.

7.3.2. SENSITIVITY ANALYSIS OF OPTIMISATION SETTINGS

The optimisation settings can be adjusted to influence the search for Pareto optimal solutions. In this section the effect of these settings on the results (in this case mainly the Pareto front) are discussed. This is limited to the MOO trading off lead time and investment cost.

Target front size
The desired total amount of Pareto optimal solutions influences the optimisation. In Figure 7.9 the generated Pareto front at varying target front sizes are displayed. With a low target front size the optimisation meets the convergence criteria earlier and does not assess as many solutions as with a larger target front size. Therefore a MOO with a low target front size requires less computational time but as can be seen it misses many Pareto Optimal solutions found in MOOs with a larger target front size.

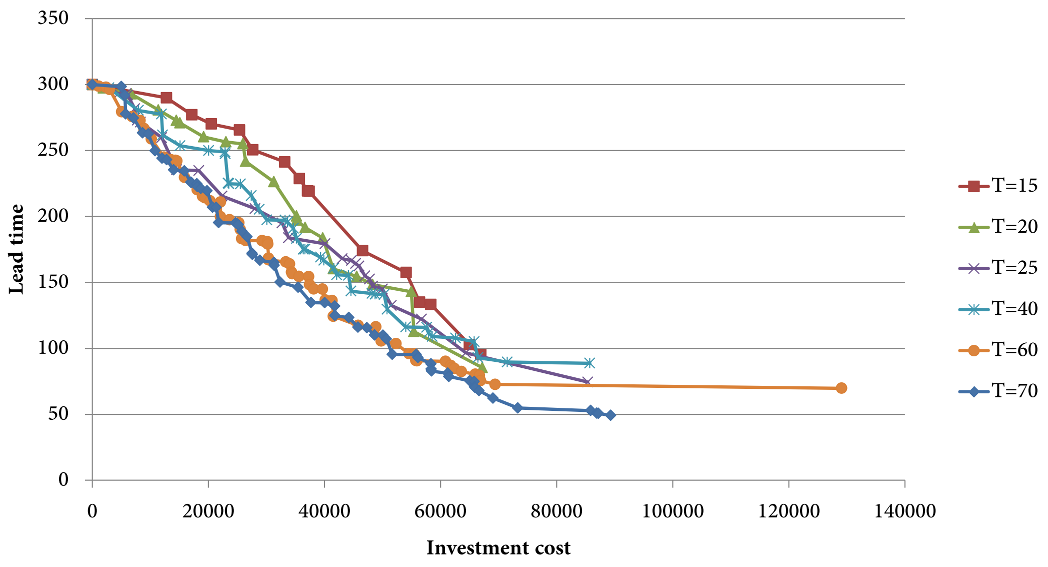


Figure 7.9: Pareto plots using the proposed method using various target front sizes

It can be seen that with an increasing target front size the Pareto front shape remains more constant. Therefore it is important to select a target front size with a sufficiently high value for the problem at hand upon using this optimisation method. Insufficient data is available to determine a relation between the number of parameters in the design vector and the minimal target front size to yield consistent results.

Start population
The optimisation method combines populations to create an offspring population (see Appendix C). Therefore the start population influences the results. In Figure 7.10 three Pareto fronts can be identified. The first is based on a starting population with all activities at a level of automation equal to the lower bound, the second with intermediate levels of automation (level 2 and 3 alternately) and the third with all activities at a level of automation equal to its upper bound. These three settings are refered to as "Start = low", "Start = intermediate" and "Start = high" in the legend of Figure 7.10 respectively.

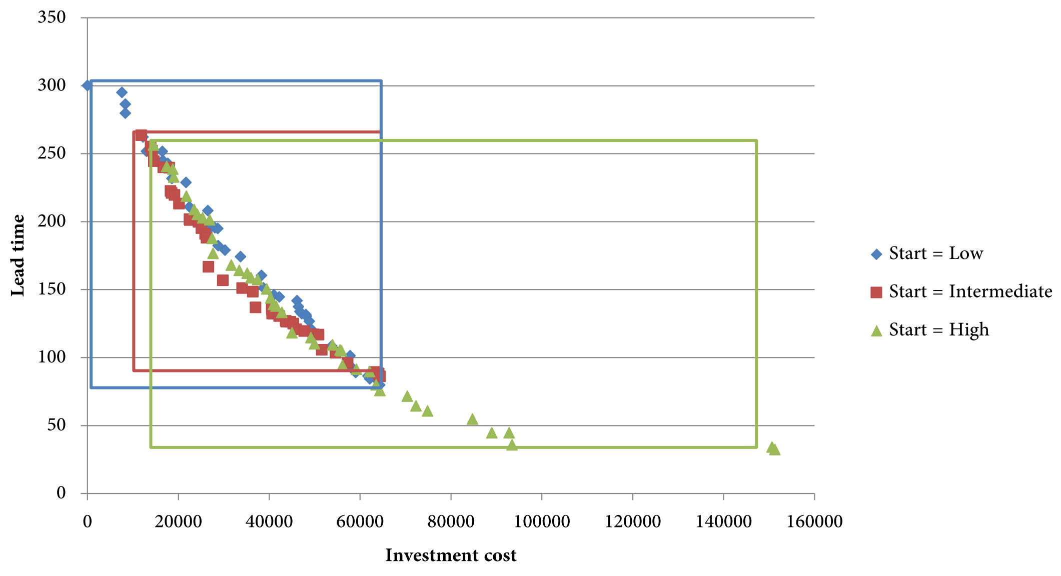


Figure 7.10: Pareto plots for the optimisation using different start populations

In Figure 7.10 it can be seen that the population influences the results. The starting populations with low, intermediate an high levels of automation are more successful in finding Pareto optimal solutions in the regions of low, intermediate and high investment cost respectively. This is in accordance with the expectations since this method uses the start population and generates new populations partially based on them.
It can be concluded that the start population influences the found Pareto front. The Pareto fronts find solutions in the same order of magnitude, but depending on the start population outliers are found. Hence if for an optimisation problem the "low lead time - high investment cost" solutions are of particular interest, then a start population with all variables set at their upper bound would provide more results in this region of interest.

Weighting factor

The weighting factor influences the change in the variables from selected previous experiments. The value should range between 0,5 and 1,0 according to the algorithm theoretical documentation [53].

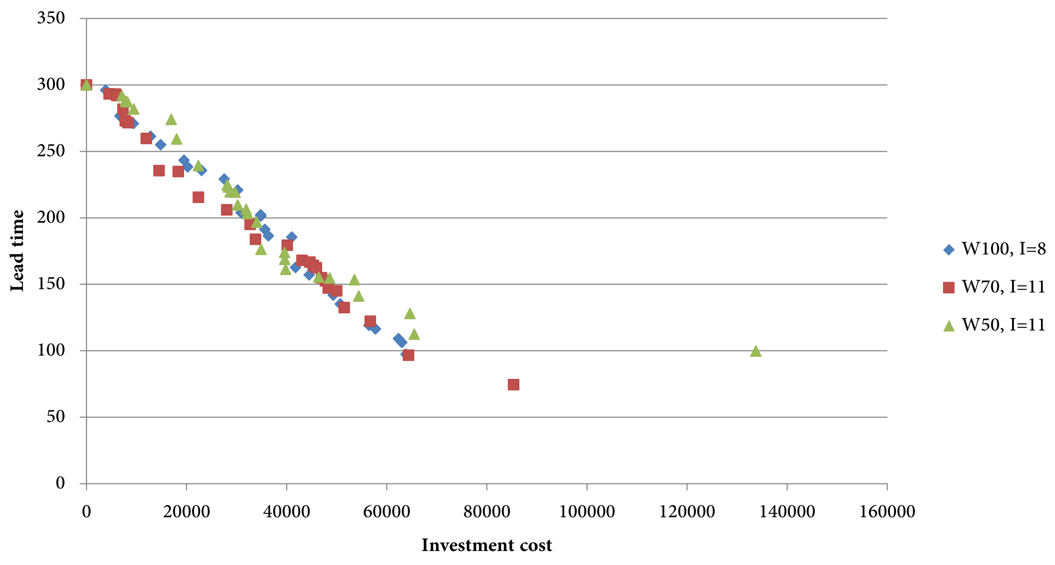


Figure 7.11: Pareto plots using the proposed method using various weighting factors

In Figure 7.11 the same optimisation is performed for different weighting factors. In the legende W100, W70 and W50 use a weighting factor of 1,0 0,7 and 0,5 respectively. The I in the legend indicates the total amount of iterations before the solutions converged. No significant distinctions can be observed in the resulting Pareto fronts other than the outlier Pareto optimal solution for a weighting factor of 0,50. Since no significant deviations could be discovered for the different settings the default setting of 0,7 is used in the remainder of this research.

Inverse crossover probability
The inverse crossover probability influence the chance that in the next generation unadjusted genes are used. The value can range between 0 and 1. With a low inverse probability more genes are reused in subsequent iterations. A low value hence leads to a higher probability of getting trapped in a local optimum. On the other hand however a very high value might lead to not reusing "healthy genes", therefore a intermediate value is often suggested.

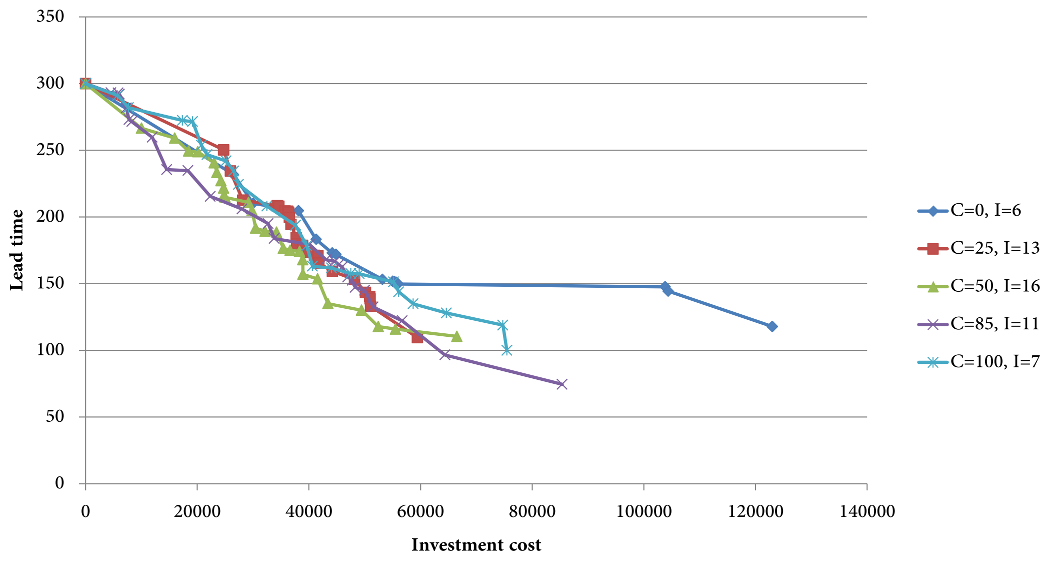


Figure 7.12: Pareto plots using the proposed method for varying inverse crossover probability

In Figure 7.12 the effect of different settings for the inverse crossover probability is illustrated. It can be seen that the settings have a distinct influence on the results. The lower values of the inverse crossover probability lead to more clustered Pareto optimal solutions. At higher values of the crossover probability the Pareto optimal solutions are located more scatter with an optimum at 0,85. The Pareto front for the inverse crossover probability of 0,50 is located the closest to the origin. It is expected that this is not necessarily due to the crossover probability but due to higher amount of iterations required in this run. This should be investigated more thorough however.
In the remainder of this research the value of 0,85 is used for the inverse crossover probability for its coverage of a large part of the design space and relatively low amount of iterations required to generate this solution.

7.3.3. Optimisation consistency
From the sensitivity analysis in the previous chapter it can be concluded that the optimisation settings influence the results. It is of importance that the optimisation eventually is reliable and hence that it yields consistent and stable results. This section discusses the stability of optimisation without adjusting the settings. Hence it investigates if the results of an optimisation, using identical settings, yield results with reliability and uniformity. This provides information to select the settings for a reliable optimisation in future applications.
In the previous section it became apparent that the results are especially sensitive for the settings of the target front size. A higher value of the target front yields more iterations and a Pareto front with, on average, more solutions closer to the origin (and hence more optimal) than optimisation with a smaller target front.
For this research a consistent and stable solution is desired and hence it is of interest at what settings the results display consistent behaviour. Therefore the optimisation is executed multiple times with identical

settings. This process is executed for three different values of the target front size. The results of the experiments are displayed in Figure 7.13. It can be seen that with an increasing target front size more similar solutions are found and hence more similarity is expected between the results.

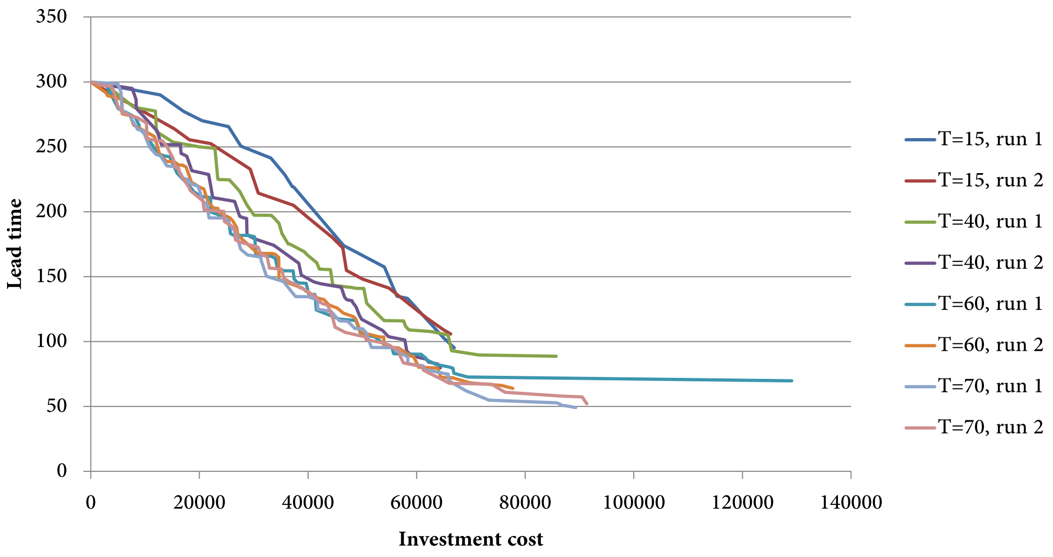


Figure 7.13: Pareto plots using the proposed optimisation for different target front sizes. For each target front size the experiment is twice to investigate consistency.

Upon closer inspection of the design vector of the Pareto optimal solutions it is seen that the process architectures on the two Pareto fronts do differ however. Due to the large amount of options for automation (4^{32} or $1,8 \cdot 10^{19}$ in this example) of which multiple automation initiatives have similar effects (e.g. automation of a specific task might have a similar impact on lead time and cost as another task) the optimisation exchanges these solutions on the Pareto front. By investigation of the actual design vectors of the Pareto optimal solution of the two fronts in Figure 7.13 no identical design vectors are found.
A profile of the Pareto front with corresponding design vectors is provided in Appendix E. This dataset has been enhanced to provide information on the found solutions. From this information it can be seen that the solutions on the Pareto fronts show similarities in the selection of activities to first optimise. Differences are still present however, especially at the solutions with a lower investment cost. It is expected that reliability increases with target front size. For 100% identical Pareto fronts and hence complete reliability an exhaustive search is required.

7.3.4. Optimisation convergence

Besides the consistency of the optimisation it obviously is also of high importance that eventually the actual optimal solutions are found. To verify that the optimisation algorithm is able to select the optimal design vectors (i.e. process architectures), optimisation results are validated with data of an exhaustive search. As explained in Section 5.3 an exhaustive search quickly becomes computationally expensive and therefore a case with specific constraints is used with an exhaustive search time of less than two hours. Specifications of the configured process can be found in Appendix F
The exhaustive search consists of 73728 feasible solutions, taking 119:35 minutes[1]. Upon inspection of the front, 52 Pareto optimal solutions are identified. The same process is also optimised using the optimisation algorithm. The settings of the optimisation are shown in Table F.4. This resulted in 11 iterations with in total 520 experiments performed in 17:22 minutes leading to 50 Pareto optimal points. Both the results of the exhaustive search and the optimisation are illustrated in Figure 7.14.

Based on these results it can be concluded that the actual Pareto based on an exhaustive search is closely approximated by the solution of the optimiser using a suitable large target front size.

[1]Using a i5-3380M CPU @ 2.90 GHz

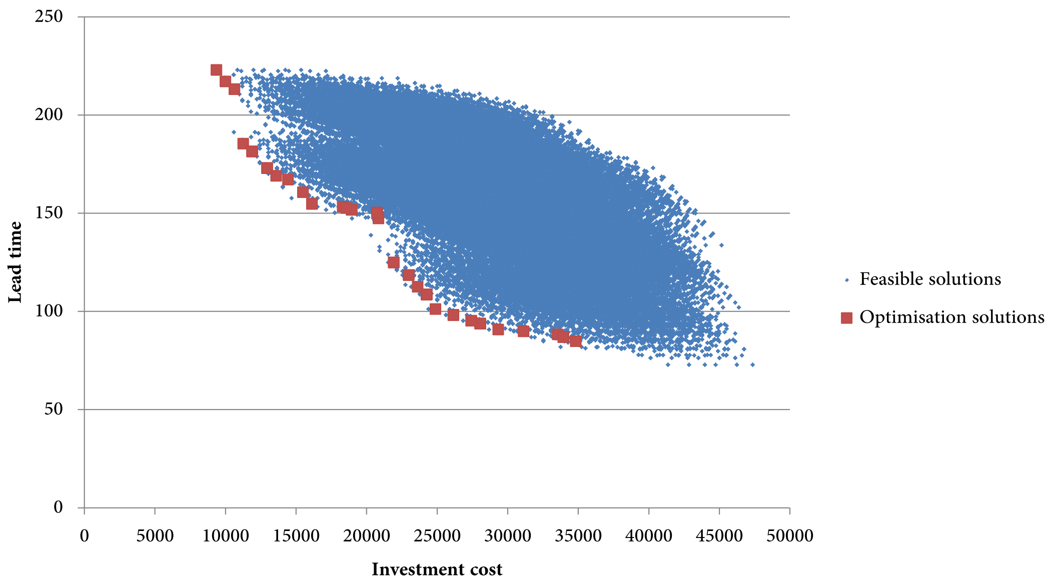


Figure 7.14: Pareto plot with results of an exhaustive search and results of an optimisation using the proposed optimisation method

7.4. DISCUSSION ON METHODOLOGY VALIDATION

In work of Smith and Morrow [54] the validity of Product Development Process (PDP) models is discussed. According to them four levels of validity of PDP models can be identified. Firstly the "face" validity, secondly validity with retrospective data, thirdly by using it in an experimental setting and finally by using it in the real world. Based on this classification of validity the methodology in this research is discussed in this section.

According to the described criteria the proposed methodology has "face" validity. The proposed methodology and all underlying assumptions, theories and outcomes of test cases have been discussed with experts and according to their judgement the methodology is valid. It should be noted that the amount of experts consulted for validation is limited and therefore it is suggested to consult an additional range of experts for validation of the generated results in the next chapter (Chapter 8).

Smith and Morrow define the next level of validation as the application of the methodology on existing but retrospective data in industry. The case study in the following chapter shows that the methodology is successfully applied. But the validation is incomplete since no full set of retrospective data is available. Acquiring such a set of data is a challenge since it would require information on many different automation initiatives. Of these initiatives information such as the development cost (split up in multiple cost attributes) and their effect on total lead time would be required. The author is uncertain if such a dataset is available and secondly if it is publicly available.

Therefore the third level of validity of Smith and Morrow is deemed more suitable for the validation of this methodology. Therefore an experimental set-up is required in which a process is modelled consisting of the PDP specific behaviour. In this experiment the different activities should be automated at different levels of automation and subsequently the process should be executed using different combinations of the automated activities. Also the effort required to automate different activities should be monitored to be able to validate the cost estimation of the proposed methodology. It should be noted that the proposed experiment is expensive in terms of required (human) resources.

8
CASE STUDY: RUDDER HINGE CONNECTIONS

In the previous chapters the complete methodology has been explained and all different elements have been discussed in detail. The framework has been verified as discussed in the previous chapter (Chapter 7). This chapter will present the capabilities of the model by applying it to a real Product Development Process (PDP) of the hinge connections of a rudder of a business-jet sized aircraft. Due to non-disclosure agreements the inputs and outputs in this chapter have been normalised.
First in Section 8.1 the case study is introduced by discussing the product and process. Next, the in Section 8.2, the inputs used in the case study are discussed. Following this, the results op the optimisation are presented in Section 8.3. A discussion on these results can be found in Section 8.4. Analysis of different automation scenarios are discussed in Section 8.5. Finally, in Section 8.6, the sensitivity of the input parameters is discussed.

8.1. HINGE CONNECTION DESIGN PROCESS

The proposed methodology and optimization framework is applied to an industrial PDP. The selected case is the conceptual design process of the hinge connections of a rudder assembly on the vertical tail of a business jet aircraft. This case provides a representative case of complex PDP, featuring many interesting characteristics: iterative loops resulting in rework, many involved departments and a mix of creative and repetitive tasks to name just a few.

8.1.1. PRODUCT DESCRIPTION

Since not every reader is familiar with the concept of a hinge connection the product under consideration in this case is briefly discussed. A hinge connection is a movable mechanism connecting multiple objects. The hinge connection in this research is the interface between the Vertical Tailplane (VT) and the rudder. This interface, depending on aircraft size, consists of multiple hinges which are aligned on a virtual hinge line. Typically 4-6 hinges are present for the VT and rudder interface. Also hinges to connect the control mechanism (actuator for fly-by-wire) are present but these hinges are out-of-scope for this research.

An example of a hinge can be seen in Figures 8.1 and 8.2. Different elements can be identified such as a bolt, sleeves, bushes, clevis lugs, center lug, nut, bearing and a locking mechanism. Furthermore in the isometric view two brackets can be seen, these brackets are out of scope. Solely the coloured section in the isometric view of Figure 8.1 is part of the design scope of this research.

8.1.2. PROCESS DESCRIPTION

The design of the hinge connections is a sub-process of the total PDP of a rudder. The process of the design of the hinge connections can be subdivided in the conceptual design and the detailed design. In terms of man-hours of work the conceptual design requires around 60% of the work and the detailed design around 40%. Due to the iterative nature of the conceptual design and labour intensive work with strong interaction between departments this phase was selected as the process scope of this research.

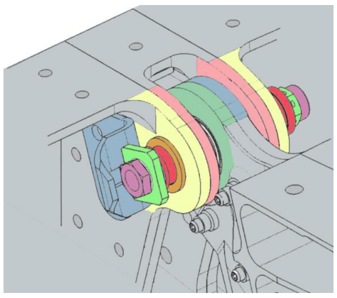


Figure 8.1: Indication of scope of the hinge in this research (courtesy of Fokker Aerostructures)

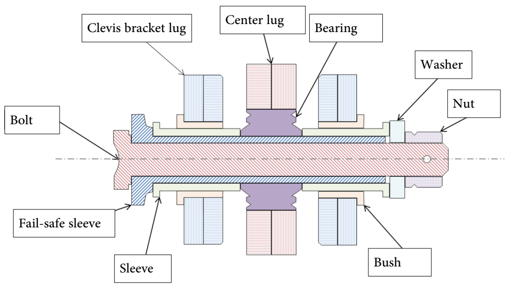


Figure 8.2: Section of a typical fail-safe hinge as used in rudder designs (courtesy of KE-works)

The conceptual design phase consists of 14 tasks decomposed into 65 activities. The involved departments are stress engineering, design engineering, cost engineering and weight engineering. In total eight resources are involved, ranging from cost engineers to the project manager. An overview of the different tasks and iterative loops in this process can be seen in Figure 8.3. Here it can be seen that two iterative loops are present. The first loop encountered during the process is due to unplanned iteration if the conceptual design fails to meet the requirements regarding stress and geometrical constraints. The second iterative loop is encountered if stress and geometrical requirements are met but when cost and/or weight requirements are not met. The different resources involved in the process are indicated by means of the numbers below the tasks.

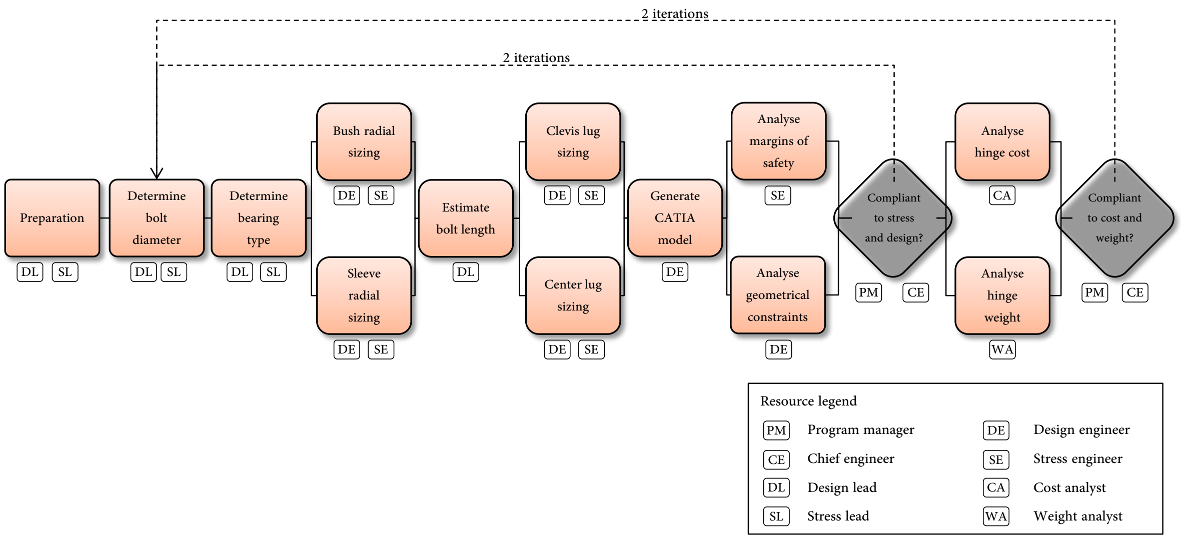


Figure 8.3: Flowchart of the conceptual design phase of the hinge connections on task level

8.2. CASE STUDY INPUT PARAMETERS

As discussed in Chapter 3 this methodology requires multiple input parameters besides the design vector. These input parameters are constant throughout the process of optimisation. The input parameters can roughly be split up in 4 parameter sets:

- Simulation settings: settings to determine the behaviour of the simulation.
- Methodology parameters: parameters used by the methods for the calculation of time durations and costs.
- Resource parameters: parameters used to model the resources.
- Process parameters: parameters to model the process on an activity level.

These input parameter sets will be discussed individually in the following paragraphs to provide qualitative information on the inputs used in this case study. Unfortunately quantitative can not be made publicly available for confidentiality reasons.

Simulation settings
The simulation is using specific settings for the process at hand. The resource costs of the developers and knowledge engineers are similar to those of KE-works employees. Furthermore no shared server and Workflow Management Software (WMS) license are assumed to be present in the current state. Initial process cost (required for Return on Investment (ROI) calculations) are set to the process cost as calculated by the simulator using the current process architecture. In Appendix G in Table G.1 an indicative overview is provided of the input parameters of the simulation settings.

Methodology parameters
This is the first case study on which the methodology is applied. The Cost Matrices and Duration Matrix (DM) as discussed in Chapter 3 are developed for this case study.
As discussed in Section 3.4 in this research the DM is determined by performing a dedicated workshop. This workshop is dedicated to this case study and therefore the process in the workshop consisted of similar activities as the activities performed in the PDP in this case study. Three response groups with varying levels of expertise participated. Each group consists of four participants of different departments within Fokker Aerostructures. In total 18 experiments were conducted at varying levels of automation. Although this workshop already required a substantial amount of time from experts, the author recommends a more detailed workshop with better time-tracking on activity granularity level to increase relevance of the DM.
The Cost Matrixs (CMs) used in this case study are based on expert judgement in combination with empirical data of previous projects executed by KE-works. $CM_{KA_{int}}$ and $CM_{KA_{ext}}$ are exceptions and the values in these matrices are based on research of Milton [50].

Resource parameters
During an interview with the lead designer of different hinge connection designs the resources assigned to a typical hinge connection PDP were discussed. This includes the amount of available resources and their wages. Currently in the process only one resource is used for the lug sizing. From the flowchart in Figure 8.3 it can be seen that these tasks could be performed in parallel if sufficient resources are available. It is decided to limit the resource to one (and hence no parallel processing of the lug sizing is possible) due to cost implications. In this case the company rather has one resource fully utilised on one project than two resources partially utilised on the project.

Process parameters
The process as can be seen in Figure 8.3 is decomposed to an activity level. During multiple interviews with experts, familiar with the design process of hinge connections, these activities and their relative durations were formalised. Of all activities the required resources are determined. Furthermore the process is modelled in a Design Structure Matrix (DSM) to correctly model the precedence constraints.
For each activity the current level of automation and the maximum level of automation is determined. The current level of automation is determined using the criteria discussed in Section 3.3. The default maximum level of automation is set at a level of automation of 4. If due to complexity or any other reason the automation level is limited the maximum level of automation is capped. This is for example the case with the decision at the task to verify the compliance with cost and weight. The activity of the decision making itself involves too many tacit and human procedures, furthermore the expert expressed that human control is desired for this activity. The remainder of the activities in this task can be automated however to support the decision making process.
The Knowledge Engineer estimated per activity the required development effort in hours. This estimate was verified with a former estimate of KE-works regarding development effort in automation of hinge connection design.
Furthermore information was elicitated on the required applications per activity. Of these applications the integration class is determined and this information is submitted in the dedicated Excel configuration file.

8.3. OPTIMISATION RESULTS

The simulation of the current process architecture results in a total process time less than 5% different from the total process time estimated by the experts during interviews. While the estimation provided by the sim-

ulation framework is a bottom-up estimation, the one provided by the experts was top-down. The simulation thus matches with the expected total process time. Comparison of process lead time is not discussed since the methodology does not take into account the rhythm of working hours of human resources.
A Multi-Objective Optimisation (MOO) is applied to the process with inputs as described in Section 8.2. The result of the MOO can be seen in Figure 8.4, here a clear Pareto front can be identified. For this optimisation in total 5560 experiments were performed during 70 iterations leading to a total of 62 Pareto optimal solutions. Details of the optimisation settings can be found in Appendix I in Table I.1. This optimisation will be referred to as optimisation 1 from now on.

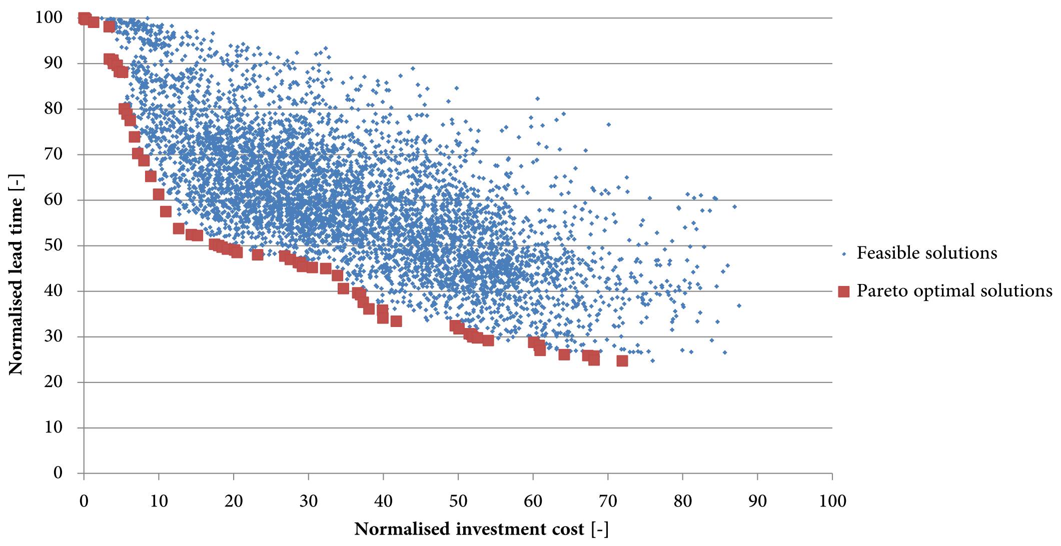


Figure 8.4: Normalised Pareto plot for the MOO trading off the lead time and the investment cost (optimisation 1)

The MOO in Figure 8.4 represents a trade-off between lead time and investment cost. This provides the user with information on the investment of specific process architectures and corresponding lead time reductions. As previously discussed in Section 2.5.4, automation potentially leads to a reduction in process cost. The lead time and investment cost are marginally influenced by the process cost and therefore the process cost is not substantially taken into account in the MOO, trading of the lead time and investment cost. For companies the ROI is often of the highest interest since this provides a quantitative business case. The ROI is the ratio between the investment and the benefits of this investment. In this chapter the investment is equal to the calculated investment cost and the benefit is equal to the reduced process cost. It should be noted that as discussed in Section 2.5.4 multiple other advantages and disadvantages are involved with the application of automation.
To take into account the ROI in the optimisation a different objective function needs to be defined. The general formula to calculate the ROI is stated in Equation 8.1. In this research the gain from the investment is the product of the reduction in process cost and the amount of times the project is executed using the automation initiatives. Hence the *ROI* and the Gain from investment are both unknown. The formula is therefore rewritten to Equation 8.2

$$ROI = \frac{\text{Gain from investment - Cost of investment}}{Cost of investment} \tag{8.1}$$

$$\text{Number of projects until BEP} = \frac{\text{Cost of investment}}{\text{Gain from investment per project}} \tag{8.2}$$

In Equation 8.2 the minimum amount of projects (i.e. number of PDP cycles) to achieve the Break Even Point (BEP) is calculated. This is one of the objective functions for the MOO of which the Pareto front is provided in Figure 8.5. The other objective function is the project lead time. For this MOO in total 5560 experiments are performed over 70 iterations leading to a total of 31 Pareto optimal solutions. Details of the optimisation settings can be found in Appendix I in Table I.2. This optimisation will be referred to as optimisation 2 from now on.

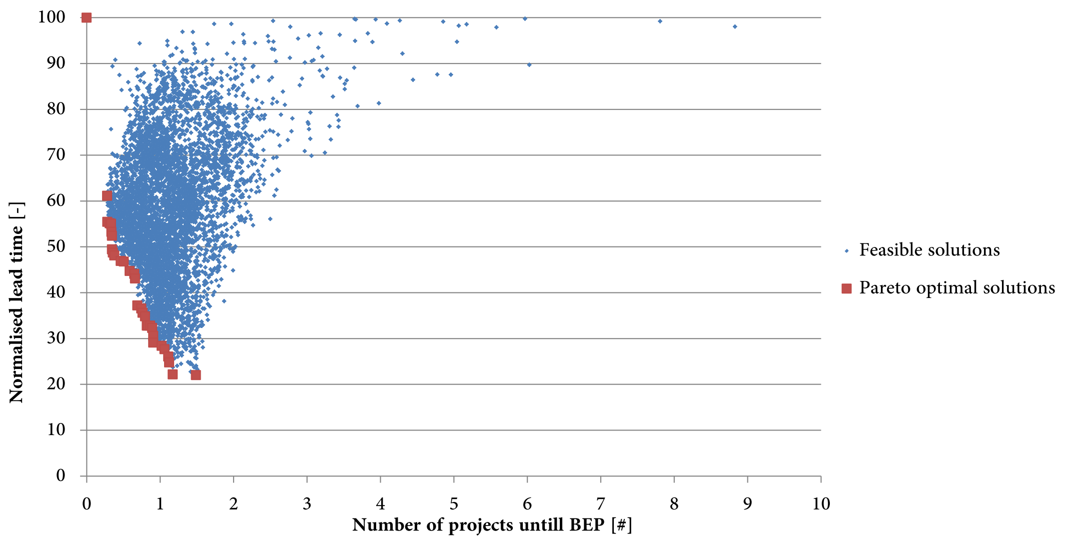


Figure 8.5: Normalised Pareto plot for the MOO trading off the lead time and the number of projects until BEP (optimisation 2)

The graph in Figure 8.5 provides the user with more information to generate a business case for the implementation of automation. It differentiates itself from the Pareto front in Figure 8.4 since it takes into account the (reduced) process cost. Each point in this graph represent a single process architecture. The process architectures in this graph differ in the individual levels of automation of the activities.
In the following section the results as presented in Figures 8.4 and 8.5 will be discussed in more detail.

8.4. DISCUSSION OF OPTIMISATION RESULTS

The results as presented in Section 8.3 contain a lot of information but need further analysis. It is of interest to investigate the different Pareto optimal solutions and their corresponding design vectors to assess trends in the optimisation and differences between the two MOOs as described in the previous section. First the two Pareto fronts will be discussed individually, followed by a brief comparison.

Optimisation 1: Lead time - Investment cost

In the graph of Figure 8.4 a Pareto front with different slopes can be identified. Hence it can be concluded that some automation initiatives lead to a bigger change in lead time reduction for a lower additional investment than others. Furthermore it can be seen that the density of Pareto optimal solutions is higher at the top left and middle of the curve. This does not necessarily mean that in this region more Pareto optimal solutions are present but apparently the algorithm finds most of the solutions here. Furthermore in the graph a gap can be seen in the Pareto front around a normalised investment cost of about 45. Upon analysis of the results this gap can be explained by the costly automation initiative for the analyse activity of the task *analyse hinge weight*. Automation of this task requires a significant amount of development and therefore causes a large jump in investment cost. The investment of a little less than 7% resulted in an insignificant increase in lead time. This clearly shows the use of this methodology, one is able to objectively assess the costs and benefits of a specific automation initiative.
In Appendix H an overview of the levels of automation per activity for all Pareto optimal solutions is provided. The levels of automation are indicated with colours to enhance the image to identify patterns. A simplified graph based on this data, generalised to task level can be seen in Figure 8.6.

It can be seen that the activities belonging to the tasks of lug sizing are fully automated for most of the Pareto optimal solutions with a low lead time. This can be explained due to the high amount of completions of these activities. These activities are executed for all hinges and multiple times due to the iterative loops. Therefore these activities are quickly automated to the highest level of automation.
Besides, it can also be seen that specific activities are not automated at all or solely in the high investment cost region. These activities are mainly of the analyse activities. This can be explained since these tasks often require a higher investment due to development and integration costs. An exception is the acquire activity of the *preparation* task, this can be explained due to the high development cost of this specific activity.

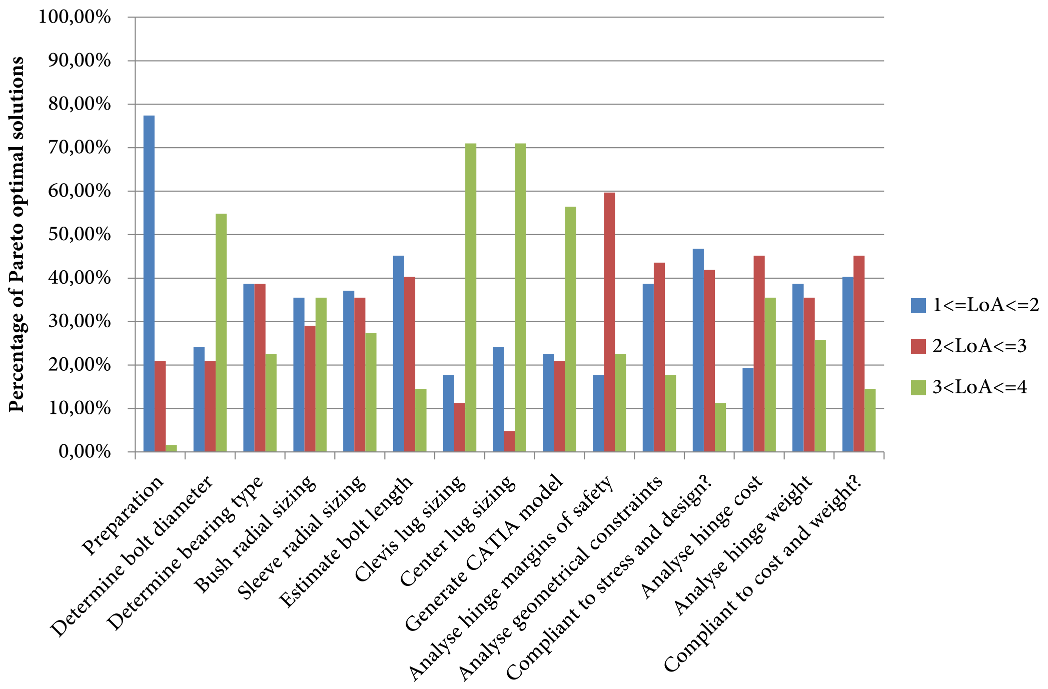


Figure 8.6: Simplified overview of the average level of automation on a task level for all Pareto optimal solutions of optimisation 1

Furthermore, based on the detail information in Appendix H, it can be seen that for the low investment cost Pareto optimal solutions the highest level of automation is not applied. This is due to the high impact of the license cost on total investment cost. Upon closer inspection it can be seen that in this region of Pareto solutions mainly the information flow automation is applied (i.e. acquire, process and implement activities are automated).

Optimisation 2: Lead time - Number of projects until BEP

Please note that the value on the x-axis of Figure 8.5 is based on Equation 8.2 and therefore allows for non-discrete values. In reality it is not possible to perform 0,34 projects and for realistic results the values should be rounded-up to integer values. This would however flatten the results to a few single lines and therefore decimal values are used.

The shape of the graph in Figure 8.5 differs significantly from the graph in Figure 8.4. The experiments are more clustered with very few outliers. For a larger reduction in lead time apparently more projects need to be executed before BEP.

A gap can be seen around a normalised lead time of 40. In Appendix H an overview of all design vectors for all Pareto optimal solutions is provided. This overview provides the opportunity inspect the two specific Pareto optimal solutions at the gap. It can be seen that the optimiser chooses a radical change in the design vector. In total 24 activities are lowered in their level of automation and 9 activities are increased. Hence more than half of the variables in the design vector is adjusted around this gap. The lower level of automation of 24 activities is required to compensate for the high cost of increasing the other 9 activities in their level of automation. In particular the analyse activity of the *generate CATIA model*. Automating this task is a costly exercise. It can be seen that for all Pareto optimal solutions below this gap this activity is fully automated, hence it only becomes an efficient investment at a higher amount of projects until BEP.

Furthermore one could say that automation shows great potential for the process in this case study. Many Pareto optimal solutions exist with a low amount of projects until BEP required. The lead time can be reduced significantly with an appealing business case. It does lead to the justified question if the model is not overestimating the ROI. Referring back to Equation 8.2, this overestimation of the ROI could be caused by either an underestimation of the cost of investment or an overestimation of the gain from investment per project (reduced process cost). To address a potential underestimation of the investment cost it is advised to re-discuss the input parameters for the tasks *Determine bolt diameter*, *Clevis lug sizing* and *Center lug sizing*.

These tasks strongly influence the lead time reduction, at relatively low costs. The potential overestimation of the gain from investment per project could be caused by the determination of process cost. This method assumes that a resource is fully occupied by the activity during the activity lead time, while in reality this might not be the case. Full automation (level of automation 4) would thus overestimate the benefit of automation. This is a potential cause for the potential over estimation but more research is required to first validate if the methodology overestimates the benefits and subsequently what causes this overestimation. In the verification phase of this research it was observed that the resource cost influenced the Pareto optimal solutions for the MOO trading off lead time and number of projects until BEP. It is of interest to see if the same holds for this case study. Upon inspection of the design vectors it can be seen that the activities of the tasks with higher resource costs (especially *compliant to cost and weight?* and *compliant to stress and design?*) are more frequently automated. This holds especially for the activities of the *compliant to stress and design?* task since it is completed multiple times due to iteration.

Comparison of MOOs

Upon comparing the two optimisations as analysed above some commonalities and differences are observed. First some commonalities. Both optimisations show that the activities with multiple completions (i.e. in iterative loops) are more effective to increase the level of automation. Furthermore it can be seen that both optimisations have specific activities of which the level of automation is increased from a specific threshold (e.g. analyse activity of *generate CATIA model* task).

Secondly some difference. Optimisation 1 takes into account resource cost in the investment cost but the contribution is insignificant on the total investment cost. Therefore the activities with higher resource cost are not treated any different than other activities. This is not the case for optimisation 2 where the resource cost is strongly reflected in the process cost and hence in the amount of project until BEP. This leads to two major differences between the optimisations. First of all in optimisation 2 more activities are fully automated because the highest level of automation gives the lowest lead time but also zero activity process cost. Secondly it can be observed that in optimisation 2 other tasks are automated, specifically the tasks with a high resource cost.

Another difference between the two optimisations is the range of solutions in terms of lead time. Optimisation 2 succeeds in finding lower lead times. This could be the result of the search algorithm; optimisation 1 seeks to minimise investment cost and therefore is less successful in finding design vectors with high investment cost and low lead times.

If the Pareto optimal solutions of both optimisations are plotted in the same chart another interesting thing is observed. This can be seen in Figures 8.7 and 8.8.

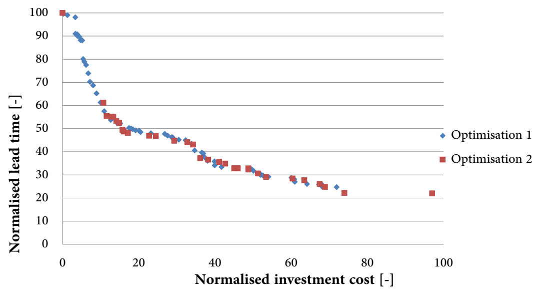


Figure 8.7: Normalised Pareto plot with Pareto optimal solutions of both optimisation 1 and 2. Plotted for lead time and investment cost

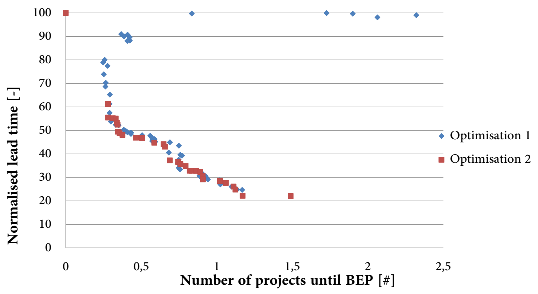


Figure 8.8: Normalised Pareto plot with Pareto optimal solutions of both optimisation 1 and 2. Plotted for lead time and number of projects until BEP

Here it can be seen that the results of optimisation 2 all are also close to the Pareto as generated by optimisation 1. Hence the Pareto optimal solutions of optimisation 2 are not just optimal for lead time vs amount of project until BEP, but also for lead time vs investment cost. The other way around it is not the same. Not all Pareto optimal solutions of optimisation 1 are on the Pareto as generated by optimisation 2. This can be explained with the help of Equation 8.2. For the solutions away from the Pareto front the investment cost is relatively low, and the gain in lead time is acceptable for that investment. But this is not reflected in Equation 8.2, there the reduced process costs are of interest. The reduction in process cost is not sufficient to become a Pareto optimal solution.

8.5. ANALYSIS OF AUTOMATION SCENARIO'S

The optimisation leads to many different Pareto optimal solution. This provides valuable information on how to incrementally optimise the PDP. It is also of interest to perform an analysis on this process of the implementation of specific automation initiatives. In this research the effect of five hypothetical but realistic scenarios are analysed and discussed in the following three sections. The results of these scenarios are summarised in the charts of Figures 8.9 and 8.10. In these figures the scenarios are compared with the Pareto optimal solutions obtained in Section 8.3.

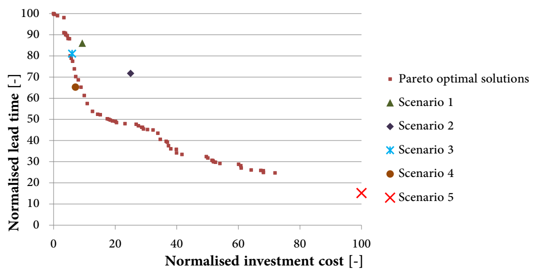


Figure 8.9: Normalised Pareto plot for the MOO trading off lead time and investment cost combined with the alternative scenarios

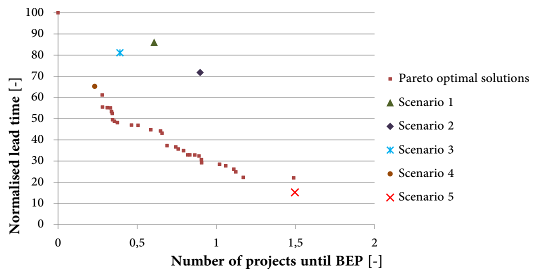


Figure 8.10: Normalised Pareto plot for the MOO trading off lead time and number of projects until BEP combined with the alternative scenarios

Table 8.1: Results of process simulation of specific automation scenarios

	Current state	**Scenario 1**	**Scenario 2**	**Scenario 3**	**Scenario 4**	**Scenario 5**
Scenario summary	Status quo	Implement KE-chain	Implement KE-chain and automate structure activities	KBE application Clevis lug sizing	KBE application Clevis and Center lug sizing	Highest level of automation
Lead time	100	86,1	71,8	81,1	56,2	15,2
Process time[1]	168,8	132,2	102,5	130,9	91,3	6,9
Time to start[2]	5,3	12,4	12,5	5,3	4,7	2,0
Process cost[3]	100	78,5	60,9	78,5	56,9	4,5
Investment cost[4]	0	9,3	25,0	6,0	7,1	100,0
KA cost	0	42	32	7	12	19
Development cost	0	0	32	6	9	52
Integration cost	0	0	0	25	22	6
License cost	0	29	11	44	38	3
Configuration cost	0	16	12	3	5	7
Training cost	0	6	6	6	6	6
Management cost	0	8	8	8	8	8
Projects until BEP	0	0,60	0,90	0,39	0,23	1,50

[1] Indexed using lead time current state is 100
[2] See footnote 1
[3] indexed using process cost current state is 100
[4] All following sub costs are measured as a percentage of scenario investment cost

8.5.1. IMPLEMENTATION OF KE-CHAIN

KE-chain is a WMS assisting the (human) resources in a process and is a good example of information flow automation. The input parameters as discussed in Section 8.2 applicable to the acquire and implement activities are based on the implementation of KE-chain. Therefore it makes sense to analyse the process performance if KE-chain is implemented. It is assumd that KE-chain only automates the acquire and implement activities in a process. This hypothetical scenario will be referred to as *Scenario 1*. KE-chain also has the potential of automating the structure steps although this would require an additional investment. This scenario is referred to as *Scenario 2*

The implementation of a WMS is simulated by adjusting the level of automation of all acquire and implement steps to become fully automated. Only the first acquire activity which should interface with external applications is not automated due to the high development costs. This procedure is also repeated for all activities improving the information quality relevance and currency (i.e. acquire, implement and structure).

Scenario 1 results in a lead time reduction of 13,9% and also the process time (i.e. total amount of man-hours) is reduced with 21,7%. The time to start experiences a major increase however. This is due to the strictness of the collaboration penalty method which was discussed in Section 4.3.3. The license cost (29%) and knowledge acquisition cost (42%) are major contributors to the total investment cost for this scenario.

Scenario 2 has a significant increase in the required number of projects until BEP. This is mainly due to the almost tripled investment cost. This steep increase in investment cost is mainly due to high knowledge acquisition, development and configuration cost. This scenario shows that the application of the proposed automation initiatives leads to a reduced lead time but not necessarily to a better business case.

8.5.2. IMPLEMENTATION OF SPECIFIC KBE APPLICATION

It can be seen in the analysis of the optimisation that automation of the block of lug sizing is effective. It would be a realistic scenario that a company is interested in developing an application fully responsible for this task. Here two scenarios are interesting to assess, first the development of a Knowledge Based Engineering (KBE) application for the clevis lug solely without automation of the center lug sizing task (Scenario 3). Secondly it is of interest to assess the effect of automation of both lug sizing tasks simultaneously (Scenario 4).

It should be noted that for realisation of these scenarios it is required that the information is provided in such a way that the acquire activity always is able to retrieve the information in the expected location and format.

Scenario 3 shows a good number for the amount of projects until BEP. The lead time is reduced significantly by automation of this single task since it is in iterative loops. The investment cost consists mainly of the integration cost (25%) and license cost (48%).

Scenario 4 shows an even better number for the amount of projects until BEP. This configuration even exceeds the found Pareto optimal solutions for both optimisation 1 and 2. This is due to the fact that the integration cost is lower for an application which has been integrated before.

8.5.3. MAXIMUM LEVEL AUTOMATION

Another scenario of interest is to investigate the effect of complete automation of the process. This means to automate all tasks to their individual maximum level of automation (as discussed in Section 8.2).

Scenario 5 shows the highest reduction in lead time but also at the maximal investment cost for this case study. The lead time is reduced by the application of automation. This results in a lower activity lead time. The cost of knowledge acquisition (19 %) and development cost (52 %) are with distance the biggest contributors to the total investment cost.

8.6. INPUT PARAMETER SENSITIVITY

In Chapter 3 it is acknowledged that it is of vital importance to use valid values in determining costs and time durations. This research requires information on 140 values used in seven matrices (CMs and DM) and 11 values for parameter settings. These parameters are expected to influence the optimisation. Therefore a sensitivity analysis of these parameters and matrices would be of great added value and would improve the relevance of this work.

Such a sensitivity study requires an assessment of the sensitivity of the Pareto front and hence multiple optimisations need to be performed to assess the sensitivity of each parameter. One can imagine that this is a costly exercise due to the large amount of parameters involved and different objective functions of interest. Performing such an extensive sensitivity study is therefore beyond the scope of this research and is recommended for future research.

9
Conclusions, limitations and recommendations

This chapter consists of three separate sections regarding the conclusions, limitations and recommendations of this research in Sections 9.1, 9.2 and 9.3 respectively.

9.1. Conclusions

Based on academic and industry sources it can be concluded that there is a strong need for a methodology to objectively quantify the costs and benefits of the application of automation initiatives, taking into account incremental automation of a process. The main objective of this research is to define a methodology to predict the effects of the implementation of automation solutions on the Product Development Process (PDP) performance by using simulation. To meet this objective, four different sub-objectives were defined in Chapter 1. The following paragraphs will reflect on these sub-objectives.
Firstly, a novel modelling framework is developed to model the PDP on an activity level. In this framework five types of activities are defined which are the building blocks of tasks. For each activity type four levels of automation are identified ranging from fully human to fully automated. The first sub-objective was to model the PDP on an activity level with varying levels of automation and hence this sub-objective was achieved.
Secondly, new methods have been developed to address the impact of changing the level of automation at an activity level on PDP performance (activity lead time, investment cost and process cost). A combination of roll-up techniques, parametric estimation and expert judgement are used for the estimations. Hence these methods successfully address the second sub-objective regarding how to model the effect of automation on the activity lead time and automation investment cost.
Thirdly, using Discrete Event Simulation (DES) a process simulator has been developed. The modelling framework and methods to estimate activity performance have been implemented in this simulation tool. Furthermore this simulator takes into account PDP specific process characteristics such as iteration, rework and collaboration. The simulation is capable of dealing with resource constraint. By means of this simulation the third sub-objective was accomplished: a model is developed to analyse the performance of specified process architectures by the use of simulation. Finally, an optimisation tool (Optimus) using a Non-dominating Sorting Genetic Algorithm (NSGA-II) is used in an integrated framework to allow for optimisation of the PDP. This optimisation finds the combination of levels of automation on an activity level leading to Pareto optimal solutions.

The developed simulation and optimisation approach have been positively verified. It can be concluded that the optimisation algorithm proofs to find Pareto optimal solutions if appropriate optimisation settings are used. Validation proves to be difficult due to scarce validation data, therefore the methodology has been validated based on expert judgement, with a positive result.
Based on the information of the verification and validation it can be concluded that it is possible to optimise a PDP by the application of different levels of automation. In this research the optimisation has been performed for lead time, investment cost and the number of projects until Break Even Point (BEP). Thereby the fourth sub-objective to develop a method to optimise the levels of automation on an activity level for multiple

objective functions is met.

The applicability and feasibility of the proposed methodology is successfully demonstrated by means of an extensive case study of a complex aerospace PDP. Two Multi-Objective Optimisations (MOOs) are performed trading of i) lead time and investment cost and ii) lead time and the number of projects until BEP. For the process under consideration it can be concluded that specific tasks can be identified showing a high potential for an increase in level of automation. Also the opposite is observed for tasks which show a low potential for an increase in the level of automation. By means of the methodology specific process architectures are identified that require a high investment, but not significantly outperform other less expensive alternatives in lead time.
On a high level the activities in iterative loops show a higher potential for automation. Furthermore the activities with a high resource cost are more efficient to automate if the Return on Investment (ROI) is taken into account.
Based on the case study it can also be concluded that the simulation tool provides great aid in assessing possible automation scenarios without the use of optimisation. The influence of the introduction of a Workflow Management Software (WMS), specific Knowledge Based Engineering (KBE) apps and full automation of the process have successfully been estimated using the simulation tool. Resulting in estimated lead time reductions of 13,9%, 43,2% and 84,8% respectively compared to the current process lead time.
Based on the previous two paragraphs it can be concluded that the feasibility and applicability of this methodology has been verified by means of representative case study and hence the fifth sub-objective has been met. Furthermore the methodology has proven to be applicable in a commercial environment and therefore all requirements are satisfied.

It can be concluded that the novel methodology is successful in objectively quantifying the costs and benefits of the application of automation. It shows great potential for the identification of optimal process architectures at various levels of automation. It has become possible to provide quantitative insight on the potential of automation in the PDP and to simulate the implementation of different automation initiatives. Hence the methodology is useful for decision makers interested in an optimal application of automation. The methodology has been applied in a single case study but due to the high customisability of the input parameters it is expected that it can be applied in many other PDPs.

9.2. Limitations

Although this research has been conducted with utmost attention and precision, it still has its shortcomings and limitations. These aspects will be discussed in this section.

• First of all, the Cost Matrices and Duration Matrix are determined based on a limited selection of available data sets, expert judgement and a dedicated workshop. These input parameters have not been validated to the desired extent however. The correctness of these parameters is essential for this methodology. The input parameters would yield a higher validity if more extensive research, using a quantitative approach, is performed.
• The second limitation is strongly related to the first limitation: although most of the process input parameters are based on quantitative data, some parameters (e.g. development effort) are based on expert judgement and hence prone to over- or underestimation. Therefore some parameters strongly hinge on the experience and judgement of the knowledge engineer. Absence of an experienced knowledge engineer would be a limiting factor for the application of this methodology. More formalised guidelines for the estimation of these parameters could be developed, resulting in more consistent process parameter identification, independent of the user.
• Third, the applicability of this methodology has been proven for a single case study. More research is required to determine generalisability of this methodology. Hence currently only a limited applicability is assured.
• Moreover the lead time of a process is determined without taking into account the working hours of human resources. This leads to optimistic results in terms of lead time for processes. It is also expected that if working hours are taken into account the effect of automation on lead time will be reinforced. Fully automated tasks do not require sleep or rest and therefore have 24 working hours per day.
• Finally the model does not take into account the Learning Curve (LC). It is expected that this leads to more

pessimistic duration estimations for non-fully-automated tasks and hence it reinforces the estimated benefits of the application of automation. If the LC would be taken into account, activities with multiple iterations would yield a lower lead time after the first completion in the non-fully-automated cases. Fully automated tasks often do not show the behaviour of the learning curve. It is therefore expected that if the LC is taken into account, that automation would become less attractive compared to the implementation as discussed in this research.

9.3. RECOMMENDATIONS

This research is a first step into the quantification of costs and benefits of design automation in the PDP. Ample research opportunities are present to improve and extend the research at hand. In this section a selection of recommendations will be discussed. These recommendations are clustered according to six themes.

Process modelling and analysis
In the developed process and analysis model solely deterministic values are used (apart from the optimisation algorithm). In reality however the PDP is not a deterministic process and consists of stochastic variables. For example the duration of activities, amount of iterations and probability of rework all have a stochastic nature. To enhance the results of the analysis of a PDP, it is recommended to account for these stochastic events. By means of this stochastic approach one would be able to assess for example schedule risk as discussed by Browning and Eppinger [19].
Another recommendation to improve the process modelling and analysis is to include the learning curve as discussed in the limitations. The learning curve reduces the lead time of an activity for each consecutive time it is performed. This recommendation can be implemented by adjusting the lead time calculation method and take into account the amount of completions of a specific activity.
As discussed in Section 4.3.3, the method used to determine the collaboration penalty is strict. Meaning that it often applies a penalty if in reality this would not necessary apply. It is recommended to extend the method to determine a certain threshold for the penalty to be applied. It is expected that this would yield results that better suit the PDP behaviour.

Methodology validity
It has been stated before that the validity of this methodology is currently solely based on expert judgement. Four topics are suggested to improve the validity of this methodology.
First, as discussed in the previous section (Section 9.2), the input parameters for the Cost and Duration Matrices require more extensive research. For the Duration Matrix it is recommended to perform a more extensive workshop compared to the workshop perform in this research. During this extended workshop it is of importance to structure the process as such that all activities can independently be identified. If the activities can be identified it is possible to measure the exact duration per activity for a specific level of automation. This should then be performed for different levels of automation, using multiple response groups. Furthermore the Cost Matrices should be validated and appropriately adjusted based on other data of previous projects.
Second, it is of great relevance to perform a sensitivity study on all input parameters. Based on this information the uncertainty of the generated Pareto front can be discussed. It should be noted that this sensitivity study is computationally expensive and requires a vast amount of research. It therefore recommended to first investigate the sensitivity of the Duration Matrix.
Third, it is advised to perform continue research on the selected case study. A potential overestimation of the ROI is expected but could not be verified. Additional research is required to assess the different aspects influencing the ROI.
Finally, it is recommended to apply the methodology on more PDPs in the aerospace industry. Hereby the feasibility and applicability of this methodology within this industry is demonstrated.

Methodology applicability
Due to the high customisability of the methodology it is expected that it can also serve as an objective trade-off method in industries other than the industry of the case study. To test this hypothesis it is advised to perform a case study, similar to the one in this research, in a different industry. A suitable industry would be the automotive industry for example.
Furthermore it is expected that this tool could also be used in different trade-offs than the ones discussed in this research. As explained the resource cost is taken into account. Hence an investigation of interest would

be to assess the trade-off between automation and outsourcing activities to low cost countries. This is just one example of an extended application of this methodology.

Integrated framework
As discussed in Chapter 6, the integrated framework is developed to enhance usability of the developed tools and methods in this research. Furthermore it discusses that improvements could be made in the computational performance of the framework. This paragraph discusses recommendations enhancing usability and computational performance of the integrated framework.
First it is recommended to reduce computational time of the integrated framework by eliminating the interaction with Excel for each experiment. Therefore a different Optimus workflow is required and minor adjustment to the Python code. It is expected that this adjustment will greatly improve computational time.
In the same line of thought it is advised to integrate a Genetic Algorithm (GA) in the Python code. This will reduce the dependency on the Optimus platform and hence increase the usability of this methodology for users without an Optimus license. Furthermore it is expected that this will also lower the computational time since the interfacing between Optimus, Excel and Python is not required. This effect depends on the computational performance of the implemented GA however.
To enhance the user experience it is recommended to improve the integration between the different elements. Meaning that the methodology can be used with only a single application. In the current research this has been partially achieved by using Excel to integrate Excel and Python. Integration of Excel, Python and Optimus remained unsuccessful however due to technical limitations. A platform like KE-chain could be the binding element to eliminate the technical limitations and to achieve this goal.
Finally, it was observed during the analysis of the case study results that interpretation of generated data was challenging. It is recommended to enhance the interpretation and visualisation of data. The suggested approach is to develop Visual Basic for Applications (VBA) scripts to interpret and visually enhance the generated information in Excel.

Business application
This methodology offers a whole new paradigm of quantitative process optimisation consultancy. Up till now process optimisation advice was mainly qualitative and based on experience and best-practices. One can use the elements of this research to quickly gain insight in the optimal process architectures and identify tasks with a higher gain if automated than others.
Furthermore, since the methodology differentiates between the various activities it is possible to estimate the effect of the implementation of a specific automation application. For KE-works that implies that it is possible to assess the effect of the implementation of KE-chain for any PDP. It is recommended to use this methodology in the future sales trajectory to quickly gain estimates for clients of the potential benefits of the implementation of KE-chain. This is especially the case for a pure KE-chain implementation (i.e. no add-ons or additional development) due to the low amount of required information.
Finally, it is advised to KE-works to also assess the options of integrating the simulation framework in their WMS to allow users to upfront simulate their current process. Potentially with a coupled optimiser to directly show the user the current potential for automation.

Methodology extension
As discussed in Section 1.3.3, this research has a limited scope. This research can be extended in multiple ways, of which two will be discussed in this paragraph.
The integrated framework has been developed in a modular way and can easily be extended and adjusted when needed. Process restructuring, as discussed in Section 1.3.3, has not been formally implemented in the integrated framework. The addition of process restructuring would however improve the significance of this work and is therefore recommended.
Another valuable extension would be to extend the current research for other performance indicators of importance in PDPs. One could think of integrating product quality as an objective function. Therefore additional methods are required to relate a process architecture to product quality. These methods are, to the authors knowledge, not available and thus significant research would be required. It could yield interesting results if for example sustainability of a product can be taken into account during PDP optimisation.

A
CEAS Conference Technical Paper

A technical paper based on the work performed in this research has been submitted and accepted for presentation at the 5th CEAS Air & Space conference (07 - 11 September 2015, Delft, the Netherlands). In total 200 papers will be presented by aerospace scientists and engineers from 25 different nations around the world. The integral version of the submitted paper is included in this appendix.

CHALLENGES IN EUROPEAN AEROSPACE
5TH CEAS AIR & SPACE CONFERENCE

A METHODOLOGICAL APPROACH FOR THE OPTIMISATION OF THE PRODUCT DEVELOPMENT PROCESS BY THE APPLICATION OF DESIGN AUTOMATION

*Bram Mulder**, ** (bram.mulder@ke-works.com)*
*Dr. ir. G. La Rocca***, Dr. ir. J. Schut**, Dr. ir. W.J.C. Verhagen*** (Delft University of Technology)*
**KE-works, Molengraaffsingel 12-14, 2629JD, Delft*
***Delft University of Technology, Kluyverweg 1, 2629HS, Delft*

ABSTRACT

A short lead time of the Product Development Process (PDP) is an important competitive advantage for companies. Design automation solutions provide a means to reduce the lead time and improve quality, but their development requires some investment. Before a company can commit to the development of an automation initiative, it requires an estimation of the expected costs and benefits. The objective of this research is the development of a decision support system, based on multi objective optimization techniques and Discrete Event Simulation, to evaluate the effect of introducing automation solutions in a given PDP. The system is able to generate Pareto fronts showing optimum combinations of lead time reductions versus investment cost for automation. For each of the solutions on the Pareto front, the system provides the suggested list of PDP activities to be automated and their level of automation. The system functionality has been successfully demonstrated by means of a use case concerning the PDP of an aircraft component.

1 INTRODUCTION

In the past decades a clear transition can be seen from fully human-based production techniques towards more automated systems. This transition is focused on reducing lead time, decreasing process cost and improving product consistency and quality. The same trend of adopting more automation can also be seen in the Product Development Process (PDP), driven by a growing focus on PDP improvement as a potential source of competitive advantage [3]. In particular, for many companies lead time duration is the most important performance measure of the development process, because a reduced time-to-market (i.e. lead time) results in a reduction in cost-of-delay and a larger market share [17]. Therefore a reduction in lead time is worth an investment for companies. Design Automation (DA), Knowledge Based Engineering (KBE), Artificial Intelligence (AI) and Computer Aided Design (CAD) are examples of computer based technologies adopted to improve the PDP.

Automation is in literature often regarded as a binary option for process improvements, meaning that a process either is fully automated or not at all. This is not a realistic point of view since automation can be seen as an incremental innovation. In practice it is often not possible or even desirable to automate a full process at once due to technology challenges, but also in consideration of the human side adoption of the automated solution [20]. Another practical aspect playing in favour of incremental innovation is the available budget of the company. Often concept proof of concept is generated before a whole process can be automated. Another important challenge is the continuous adjustment in processes and products, which lead to the need of extremely flexible automation solutions [24]. Furthermore it is difficult and often impossible, to predict the impact of single changes in the configuration of a process (e.g. by the introduction of automation solutions for specific tasks) of the overall PDP [7].

Before a company can commit to the development or acquisition of automation solutions, management needs critical information such as the set of PDP activities to automate first, the expected gain in lead time reduction, the cost associated to the implementation of different levels of automation or to the reconfiguration of the whole process to a specific level of automation (LoA). Figure 1 qualitatively displays the current situation where a company incrementally applies automation without knowledge of the shape of the Pareto front (i.e. the set of optimal lead time-investment cost combinations). Only a perceived Pareto front (i.e. bold guestimates based on intuition) with a high uncertainty is available. Figure 2 illustrates the desired situation in which a feasible region is known, consisting of many different PDP architectures, each one consisting of the complete sequences of process activities, with their level of automation and employed resources. Within this feasible region knowledge is available on the optimal solutions on the Pareto.

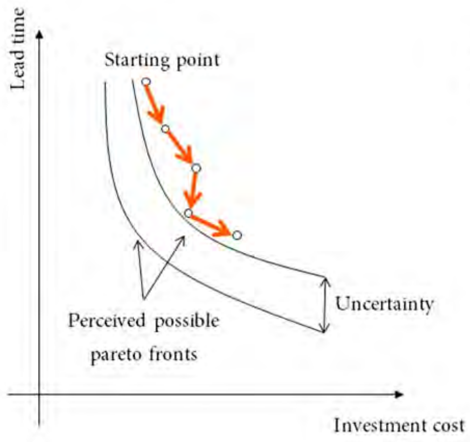


Figure 1: Current trade-off between cost and lead time in a PDP

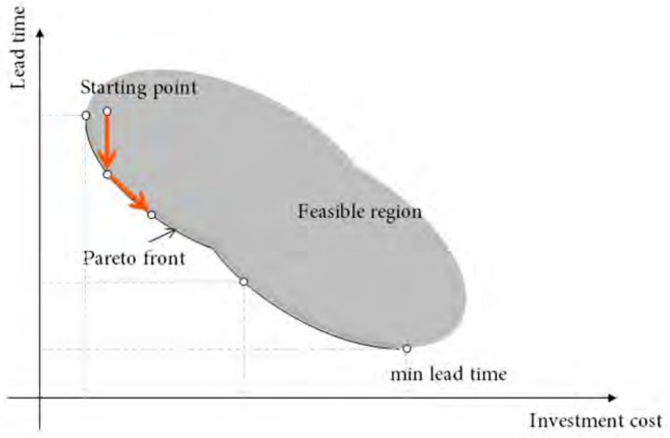


Figure 2: Improved trade-off between cost and lead time in a PDP

For companies the process architectures on the Pareto front are those of highest interest since they represent optimum combinations of lead time and required investment cost, i.e. PDP architectures for which one of the two objectives (lead time and investment cost) cannot be improved without deteriorating the other. The Pareto front can be used to estimate the investment costs necessary to achieve a certain lead time reduction, or, vice versa, the amount of lead time reduction that can be achieved with a given budget to invest in automation solutions.

Literature addresses aspects of these challenges but lacks two important aspects. Firstly, there are no models able to predict the cost and benefit of automation on the PDP performance, whilst such knowledge is of paramount importance for the management that must decide whether is convenient to invest on the development of automation solutions [25]. Secondly, the automation solutions considered in PDP literature generally do not take into account the option of selectively applying different levels of automation on different (sub)activities in the PDP.

This paper proposes a novel methodology to predict the effects in the PDP performance produced by the implementation of automation solutions. More specifically, the proposed methodology considers the complete process in the current state (hence it does not aims at restructuring it) and evaluate the influence of the application of specific automation initiatives, at single (sub) activity level, on the overall process lead time and investment cost.

To achieve this objective a new methodology is developed (i) to model any PDP as a combination of a pre-defined set of specific activities and (ii) to define different levels of automation for these activities. Subsequently, metrics are developed to measure the impact of different levels of automation on different activities, both in terms of activity lead time reduction and implementation cost. A discrete event simulator is developed which utilises the proposed process model and metrics to analyse, among others, the lead time and automation cost for the overall process (for a given process architecture). Finally, the simulator is connected to an optimiser which tries to find the most convenient level of automation for each of the PDP activities, in order to generate the Pareto front qualitatively illustrated in Figure 2.

The proposed methodology and the analysis and optimization framework are demonstrated by application to an industrial case study. The case study concerns the conceptual design phase of an aircraft component (in particular the study of the rudder-fin connection) performed by a multinational aerospace enterprise. The metrics used in this study to estimate the cost of automation (for various levels of automation) and associated lead time reduction for different type of PDP activities are based on the experience gained by KE-Works during the deployment of KE-Chain, their Workflow Management System, in various aerospace PDPs. For the discrete event simulation and optimisation, Simpy and the Optimus® toolkit are used, respectively.

2 BACKGROUND

In this field of PDP research many definitions are used and multiple viewpoints on the same topic exist. The goal of this section is to provide a clear overview and indicate how these are used in this research.

2.1 Product Development Processes

The definition of the PDP used in this article is adopted from Krishnan et al. [11]: "The product development process is considered to be a process of transformation of input information about customer needs and market opportunities into output information which corresponds to manufacturable designs, and functional tooling for volume production."

In this research the focus is on single company, multiple department projects, although the proposed methodology can be extended as necessary.

In literature the PDP is often characterised by terms like 'creative', 'iterative', 'collaborative' and 'innovative' [45, 10, 13, 6]. These PDP specific characteristics provide specific challenges which differ from those encountered, for example, in the manufacturing process. A selection of the characteristics relevant to this research is discussed in the following sub-sections.

Iteration: The cause of iteration can differ; often a distinction is made between planned and unplanned iteration [28]. Planned iterations occur when a task is attempted without a complete set of information and hence assumptions are made that need to be verified later on. Unplanned iterations occur when activities are repeated due to unexpected failure, for example due to a change in the requirements or by failing to meet a requirement.

Rework: Repeating or refining a task (i.e. rework) is a consequence of iteration. In many cases, iteration has a second order effect in terms of rework: if one task changes many subsequent tasks need to be adjusted too. Literature discusses this effect extensively and methods are proposed to quantify the probability of rework in the case of a change, and the extent of rework necessary for the whole task (e.g. is it necessary to perform the full task again or only a selection of the task activities) [27]. Also the concept of an improvement curve is discussed, meaning that the duration of a task decreases at each iteration performed by a human resource, due to the cumulated experience and increased ability [4].

Collaboration: The PDP of complex engineering products is inherently a multidisciplinary process as discussed by Reed et al. [16] in the context of aerospace industry. Multiple disciplines, often clustered in departments, need to interact and exchange information and trigger each other to start an activity. These collaborative activities have an influence on the performance of the PDP [5].

2.1.1 Process modelling of product development

Many sources in literature propose methods to model the PDP taking into account, among others, the characteristics mentioned in Section 2.1. All of these methods are based on the observation of specific behaviours which attempt to capture. For example, models are proposed to account for the overlapping of processes [23], iterative loops [21] and the dynamic and stochastic aspects of the PDP [8].

Most PDP models use an activity network as a fundamental framework [6]. Here the process is viewed as a group of related activities that work together to create a result of value [9]. The PDP is a heterogeneous process, meaning that it consists of different types of activities, each having its own characteristics. Most process models do not make a distinction about what the content of a task is (i.e. a task is not decomposed into separate and different types of activity). For an extensive review on activity network-based process model, see Browning and Ramasesh [6].

According to Browning the process architecture can be defined as the elements of process activities and their pattern of interaction [4]. This means that the process architecture not solely defines the elements of the activity itself, but also its interaction with the other activities.

The Design Structure Matrix (DSM) is frequently used to (re)structure a process by adjusting the sequence of activities, while taking into account the input/output information relation between activities [28]. Often, it is used to provide precedence constraints in simulations, but lacks the ability to efficiently describe activity parameters such as activity lead time and resource type). Thereby, in order to perform process simulation studies, DSMs are often combined with other modelling techniques, such as Business Process Modelling Notation (BPMN).

2.1.2 Key Performance Indicators

This research focuses on improvement of the PDP. Therefore it is important to define a suitable set of performance indicators and look at strategies to improve them. In literature three common Key Performance Indicators (KPI's) are based on time, cost and quality [2, 12, 16, 33]. Upon investigating these KPI's, a few interesting observations can be made. Firstly, most of the authors do not quantify the KPI's. This is in particular the case for product quality, which is often mentioned, but virtually never quantified to a measurable performance indicator during the PDP. Secondly, most authors focus on a single KPI. Even when multiple KPI's are addressed in one research, most of the optimization studies are performed on a single KPI (i.e. no multi-objective optimisation). Since the previously mentioned KPI's are very broad the KPI's used in this research are discussed in more detail in the following paragraphs.

Time: Two relevant KPI's addressed in this research are *lead time* and *process time*. Lead time defines the total time from the beginning of the project until the end and consists of process time and waiting time [2]. Process time is defined as the time a resource is occupied by an activity over the course of a process. Other examples of performance indicators used in literature are waiting time [12], iteration time and time schedule risk [4].

Cost: Product cost, process cost and development cost are just some examples of cost indicators discussed in literature. The main focus of this research is on process and investment cost. *Process cost* is the cost of the resource being occupied by a task and is a recurring cost in the process. *Investment cost* is the total cost for an investment (e.g. to develop an automation solution for a certain process activity) and is a non-recurring cost.

2.2 Automation in the PDP

Automation can have a high impact on the performance of a process. Automation is a very broad term and it is applied in many different industries and processes [40, 14]. Therefore it is important to have a clear understanding of how to see automation in the PDP in this article.

2.2.1 Definition of automation

Hart et al. [10] define automation as the ability of computer systems to perform a function without human support. Within the complex structure of any (PD)process, a number of tasks can be identified, each one implying the execution of a number of activities. In general, each one of these activities offers the opportunity to be executed with a different amount of human intervention. In other words, each activity offers the opportunity to implement a different level of automation. According to the authors, design automation is about the process of transferring domain knowledge, in the broadest sense, from the expert to a computerized system, such that the system can systematically (re)use the captured knowledge to reduce, or eliminate the human involvement in some or all of the activities involved in the

PDP. It appears that different levels of automation can be established, according to the granularity level used to decompose a design process.

2.2.2 Levels of automation

Levels of automation have been researched in different application fields [22, 17]. Several models are proposed in literature, which differ, also significantly, in the identified number of automation levels and their granularity. For example, models are proposed with a number of automation level ranging from three [1] up to ten [18]; models exist that account for the different activities that are included in a task, whilst others see a task as one block, to which one level of automation can be assigned.

A general limitation of the level of automation metrics found in literature, also of those with higher granularity, is the inability to address the collaborative aspects in the PDP. No existing model, for example, defines levels of automation for typical PDP tasks such as “triggering next step”, “storing information”, “reporting”, etc. In conclusion, none of the level of automation models available in literature was deemed suitable for the purpose of this research; thereby a new one was devised, which is elaborated in detail in section 4.2.

2.2.3 Information automation

In complex PDPs different departments are involved, often at different physical locations. These departments interact by exchanging information. To be able to work efficiently and effectively it is needed that this information is available in the right place, at the right time and in the right format [4]. Brandao and Wynn [2] estimated that 30% of the development time is spent on searching and interpreting information. This already shows that there is a lot of potential for reducing waste in this information flow by applying automation. The automation of the information flow is about improving the information relevance and currency.

The information value automation concerns the automation of the activities directly adding value to the information model of the product under development. This can be achieved by the application of computer Design Automation systems (DA). DA includes all types of dedicated computer applications ranging from tools to automate calculations (e.g. spreadsheets) to complex KBE applications able to perform generative design based on a set of input parameters.

2.2.4 Effects of automation

Automation has a lot of potential in improving the performance of the PDP in terms of lead time, process cost and product quality. However, also risks exist in the application of automation. For example, automation can negatively influence the complacency of the engineer and the situational awareness of the engineer [26]. Furthermore automation provides an easy option to generate a design and does not force the designer to think creatively for alternative solutions, hence reducing innovative and creative ideas. These effects are important to take into account upon trading off alternative levels of automation, however they are hard, if not impossible, to predict and quantify.

3 RESEARCH CONTEXT

This research follows the work performed by Schut et al. [19] and Verhagen et al. [41] into development process optimisation. The research of Schut et al. resulted in a Value Scan for process improvements by means of reducing wasted time and optimising the information flow. This method was a very high level scan of the process (i.e. at low granularity). In the succeeding research by Verhagen et al. the implementation of information flow automation in processes was assessed by using the IMPROVE method [25].

Based on this research the following methodology for the assessment of the application of automation in the Product Development Processes is proposed. This methodology consists of several steps and is visualised in Figure 3.

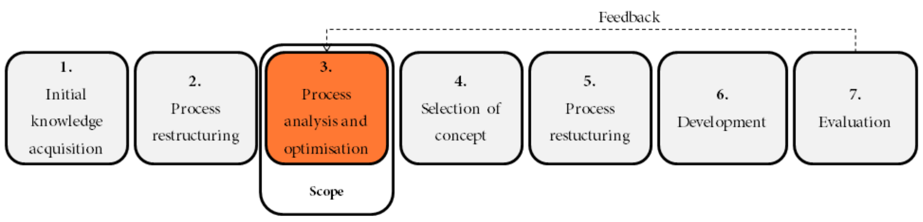


Figure 3: Steps involved in the proposed methodology

The primary focus of this research is the process analysis and optimisation. The objective is to develop a trade-off method for the optimisation of the level of automation of the product development process by using process simulation.

Optimisation of a given fixed process without restructuring proves to be of great relevance. Especially in industries with certified processes it is costly, or even impossible, to modify the structure of their PDP, hence the application of automation should be evaluated without restructuring the PDP. However, the authors are aware of the fact that a global and comprehensive optimization of the PDP cannot be achieved without considering the synergetic effect of automation solutions deployment and process structure restructuring.

4 PROPOSED MODELLING FRAMEWORK

To be able to analyse and optimise the levels of automation of activities in the PDP a framework is proposed which is able to transform process specific inputs into corresponding performance outputs, for different process architectures. In a mathematical format, this is shown in Equation 1.

$$\boldsymbol{z} = f(\boldsymbol{x}, \boldsymbol{y}) \tag{1}$$

Here $\boldsymbol{z}$ is the output vector with the relevant information (KPIs) required to make a trade-off between different options. These KPIs include, for example, lead time, automation investment cost and process cost. $\boldsymbol{x}$ is the design vector containing all the adjustable process variables, such as the levels of automation of the single process activities. $\boldsymbol{y}$ contains the input parameters used to model a given process, such as the sequence of the various activities in the process and the parameters to compute the development cost of different level of automation solutions and their lead time reduction on the process. The function transforms the design vector and input parameters into the desired output. This is illustrated in Figure 4, where also the position of the optimizer is shown, which will take care of finding the optimal vector $\boldsymbol{x}$ yielding the best $\boldsymbol{z}$.

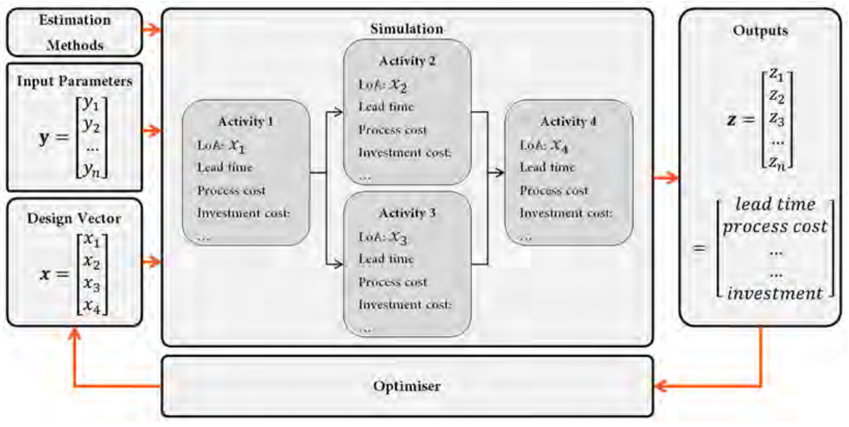

Estimation Methods
Input Parameters
$\boldsymbol{y} = \begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_n \end{bmatrix}$
Design Vector
$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$
Simulation
Activity 1
LoA: x_1
Lead time
Process cost
Investment cost
...
Activity 2
LoA: x_2
Lead time
Process cost
Investment cost
...
Activity 3
LoA: x_3
Lead time
Process cost
Investment cost
...
Activity 4
LoA: x_4
Lead time
Process cost
Investment cost
...
Outputs
$\boldsymbol{z} = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ \dots \\ z_n \end{bmatrix} = \begin{bmatrix} \text{lead time} \\ \text{process cost} \\ \dots \\ \dots \\ \text{investment} \end{bmatrix}$
Optimiser

Figure 4: Overview of the proposed PDP simulation and optimization framework

Sections 4.1 and 4.2 will elaborate on the proposed methods to model a generic process as a network of five predefined types of activity, and to define the levels of automation. The following sections (4.3, 4.4 and 4.5) describe the KPIs estimation methods used by the simulator. Chapter 5 discusses the set-up of the PDP simulation system and its implementation in the optimization framework.

4.1 Process activities modelling

A process can be modelled at different levels of granularity. The structure proposed in this research is illustrated by the example in Figure 5. Since the goal is to investigate the effect of automation on a generic process, the model decomposes any specific process in identifiable specific tasks (the first two levels in Figure 5), and, finally, each specific task into a set of generic activities. These activities can be seen as the building blocks of any PDP.

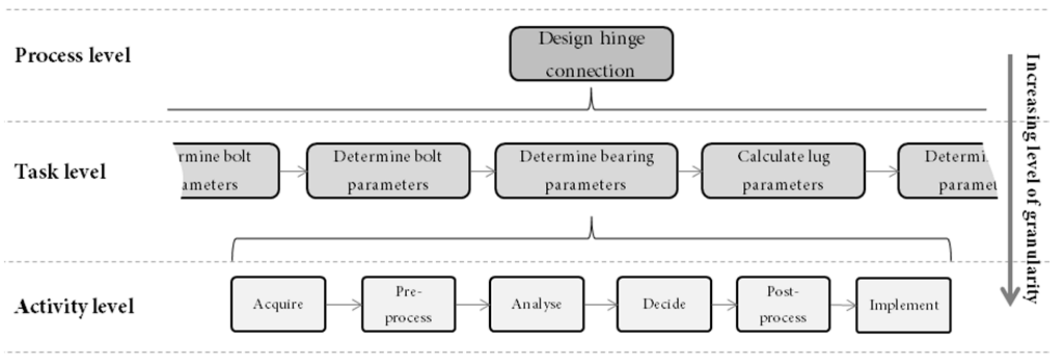


Figure 5: Decomposition from process level to activity level

Based on extensive literature research and investigation of four industrial cases, five activities were selected, from now on simply referred to as *activities*, as essential building blocks for any process. These are *Acquire*, (Pre/Post-)*Process*, *Analyse*, *Decide* and *Implement*. A comprehensive definition of these five activities is given in Table 1. It should be noted that no fixed sequence or amount of activities is prescribed for a task. As examples, one could model the Task “determining bearing parameters” in Figure 5 as a combination of the activities Acquire, Decide and Implement. The Task “calculate lug parameters” could be modelled as a combination of the activities Pre-process, Analyse and Post-process, where both the Pre and Post-process activities are of the same ’Process’ type activity described in Table 1.

Table 1: Comprehensive description of activity types

Activity type	Description
Acquire	This type of activity is concerned with acquiring all the starting conditions for a subsequent task from an external source. A starting condition is for example a trigger, knowledge or a physical product. These starting conditions are not transformed, but acquired and transmitted in the raw format they were found available.
(Pre/Post-) Process	This activity structures the information and represents it in such a way to improve the relevance of the information. In this task no information is added to the product model other than transforming the units or format of contained information. The selection and extraction of a subset of the model information is also considered a pre/post process activity.
Analyse	In an analysis activity information is transformed and new information is created. This information is added to the information model. Knowledge is used to transform the inputs into outputs. Examples include modelling, simulating, calculating.
Decide	This activity is a gateway where a decision is made with an impact on the process. At least two alternatives should be present to decide between. In this task no information is added to the product model.
Implement	This activity accounts for all the interaction with external (re)sources required to successfully continue the process. No new information is created but it is stored at a location. This task also accounts for triggering the next task.

In practice, any process of any type of complexity can be split into tasks and eventually modelled as collections of these five predefined activities. The “owner” of each task can define the given task as a sequence of (some or all of) these activities, with their relative duration distribution in the given task. For example, the Task "determining bearing parameters" of total duration TD in Figure 5 can be defined as follows: 0.60TD Acquire, 0.1TD Decide and 0.3TD Implement. By means of a Design Structure Matrix (DSM) the interaction between the activities of the various tasks can be modelled.

4.2 Levels of automation modelling

In section 2.2 the need for a modelling method for levels of automation was discussed. The proposed model is summarised in Table 2.

Table 2: Definition of levels of automation for the PDP activity types

LoA	Acquire	Process	Analyse	Decide	Implement
4	The system is the sole resource and automatically executes the activity and acquires the required information	The computer is responsible to structure the information in such a way that the next activity accepts it to be in the right format.	Computer is fully responsible for this activity. It is able to interpret the provided information and determine how to execute this activity successfully.	The computer decides and acts autonomously without interference of the human.	The computer is responsible for the correct execution of the implementation activity.
3	The activity is defined and the system suggests what to acquire and where it can be acquired. The source is responsible to acquire the items from the source. All information is available from a single source of truth.	The computer supports the user in processing the information. Hence the knowledge for processing the information is in the system but the resource needs to decide on how to apply this knowledge.	Computer supports the execution of the activity by providing tools and methods to perform calculations. Human interaction is still needed to determine intermediate steps or to verify the result.	Computer supports the execution of the activity by providing tools and methods to perform calculations. Human interaction is still needed to determine intermediate steps or to verify the result.	The human executes the implementation activity. The computer system supports the human and provides information on what to do and how to do it. System is actively involved by preventing certain actions or promoting others.
2	The activity is defined and the system suggests what items need to be acquired and where to find them. Resource is responsible to acquire items from the source.	The human is responsible to process the information. It is defined how to process the information for example by using templates.	The human is responsible for this activity and is assisted by handbook methods and procedures. The human remains the main source for the analysis.	Human is still responsible but the computer shows all relevant alternatives.	Human is responsible for the implementation activity but is supported by the system. System provides relevant information.
1	The human is the sole resource for the activity. Hence no assistance is provided by a system, manuals or procedures.	The human is responsible for processing the information. A computer or other system with basic features can be used to enhance information relevance.	Human is the only source for the methods and knowledge used in this task.	The human is responsible for the decision and the system does not provide assistance.	Human is fully responsible for the implementation activity. No assistance offered by the computer.

For each one of the activity types defined in section 4.1, four levels of automation are proposed, ranging from level 1, in which the human is the sole resource, to level 4, where full automation is provided by a computer. On the basis of the definitions provided in Table 2, each task owner in the PDP process should be able to describe the current level of automation, hence the type of resources involved in the execution of the encompassed activities.

4.3 Activity duration estimation method

The influence of automation on the activity lead time can differ per task and per activity type. This influence is captured in a coefficient accounting for the activity type and level of automation. This coefficient represents the percentage of the time a task would take, measured with the normalised time in a condition of a level 1 of automation.

$$t^* = \frac{DM_{ij^*} \cdot t}{DM_{ij}} \tag{2}$$

The coefficients are given in the Duration Matrix, DM_{ij}, and used in determining the estimated activity duration at another level of automation. In this matrix the subscript i is the task activity type (e.g. acquire) and subscript j is the level of automation. The activity lead time is calculated by using Equation 2 where t^* and j^* indicate the time and level of automation in the new case respectively.

Table 3: Example time estimation coefficients (DM)

	Level of automation			
	1	2	3	4
Acquire	100%	80%	50%	10%
Process	100%	70%	40%	20%
Analyse	100%	90%	60%	15%
Decide	100%	65%	40%	30%
Implement	100%	60%	30%	5%

Table 4: Example Knowledge Acquisition cost coefficients (CM_{KA})

	Level of automation			
	1	2	3	4
Acquire	0%	30%	70%	100%
Process	0%	40%	80%	100%
Analyse	0%	45%	90%	100%
Decide	0%	30%	60%	100%
Implement	0%	30%	70%	100%

In this research the coefficients of DM_{ij} were determined by performing a dedicated workshop. In this workshop three response groups executed the same process four times, each time using a different level of automation. The response groups varied in their level of expertise on the process used in the workshop. The values provided in Table 3 are representative but fictitious. They are different than those measured in the workshop, which cannot be published for confidentiality reasons.

The authors are aware that a correct determination of those coefficients is essential to the prediction capability of the proposed method. It will be crucial for any company that is willing to adopt the proposed method to properly estimate such values and continuously improve and update them, based on internal project knowledge.

4.4 Process cost estimation method

The process cost of an activity is based on the cost of employed resources and activity duration. If multiple resources are involved in an activity the sum of the cost per resource determines the activity process cost. In activities without human interaction (i.e. level 4 of automation) the resources are not utilised and hence the activity process costs are assumed to be zero. Here costs such as Workflow Management System licenses are considered to be an investment and are accounted for in the investment cost.

In the background section it was discussed that the use of automation also enables the use of different (e.g., cheaper) resources. No data was available on this topic and hence it is not accounted in the proposed proof of concept. In the case of iteration some models assume a certain learning curve or improvement curve, i.e. a human resource is likely to take increasingly less time to perform the same activity again and again within an iterative process. For simplification, also this effect was not taken into account in the proposed framework.

4.5 Automation investment cost estimation method

Another metric of importance in this framework is the investment cost required to automate an activity to the level of automation as stated in the design vector. In order to provide a meaningful estimation of the required investment cost, it is necessary to take into account the current level of automation and the type of activity to be automated. Software cost estimation tools assessed in literature research provided many different methods but remained solely applicable to large projects.

The cost estimation technique internally used at KE-works for knowledge engineering business was therefore adopted and modified for use in this research. This technique uses both parametric relations based on empirical data and expert judgement and a roll-up technique. Based on the analysis of several projects performed by KE-works, the development efforts required to bring a certain process activity to a target level of automation were defined. The resulting cost estimation method proposed in this research uses a similar coefficient matrix as the one described in the section 4.3. For each type of activity, and for each delta in level of automation, a cost coefficient value was determined on the basis of the technical exercises (e.g., knowledge acquisition, KBE applications development, etc.), required to produce and deploy the given automation solutions. An example of knowledge acquisition cost coefficient matrix (CM_1) is given in Table 4, which was defined on the basis of Milton [13] and adjusted with empirical data provided by KE-works. For each technical exercise a unique Cost coefficient Matrix (CM) is proposed.

Each coefficient in the matrix describes the percentage of the cost required for the full automation of the activity. For some activity types at a specific level of automation a cost attribute does not need to be taken into account, leading to a coefficient matrix with a more discrete nature (e.g. the cost of a software license purchase is only taken into account at a level 4 of automation). The method takes into account the current level of automation and estimates the extra cost to increase the current level of automation.

Some activities taken into account for the estimation of the investment cost have a reduced cost in the case of frequent use of this activity. An example is the cost estimation of the integration of an external application; this cost reduces if the same application is integrated multiple times in a project. This effect is also taken into account for license cost and server cost.

5 SIMULATION AND OPTIMISATION FRAMEWORK

The aspects discussed in the previous sections represent the main ingredients of the integrated framework for PDP analysis and optimization discussed here.

5.1 Simulation algorithm

By means of simulation the process performance is analysed in terms of lead time, process cost and investment cost. Features are implemented in the simulator to account for important PDP characteristics, such as (number of) iterations, interruption, resource constraints and waiting time.

The simulation PDP framework developed in this research is based on SimPy, a Discrete Event Simulation library, in combination with Object Oriented Programming (OOP) in Python[15]. In Discrete Event Simulation (DES) state variables only change at specified points in time, referred to as events. The simulation jumps from event to event and skips the time when no events occur and hence no state variables are changed. Furthermore DES is able to process parallel events without yielding a high computing power. The simulation ending condition is when no events are to be executed or when no event can be executed any more.

The model uses a class to model the activities in the workflow. These activities can be a regular activity (i.e. implement, process, analyse, implement) or a gateway activity (i.e. decide), from now on referred to as an *entity* and *gateway entity* respectively. In the case of a gateway entity it has a feedback loop to another entity or multiple entities. Before the simulation starts an environment is created in which the entities with corresponding properties and states are initiated based on the provided inputs. Within this environment the entities are allowed to interact by for example sending signals as will be discussed in section 5.1.2.

The entities are subject to multiple constraints. Resource constraints and precedence constraints are the main constraints. At the start of the simulation (t=0, unless otherwise defined) all entities assess if all their constraints are met; the process of assessing this is referred to as responding. Once all constraints are met the entity starts and is completed after the duration. This duration is determined based on the inputs and by means of the method explained in section 4.3. During the full duration the entity has claimed the required resources from the resource pool, hence no other entity can claim the same resource at the same time. Upon completion it interacts with other entities by sending a signal to the succeeding entity, or entities, triggering them to respond.

The adopted DES simulation algorithm distinguishes from other algorithms by the way it deals with complex PDP properties like iteration, rework and collaboration. These aspects are addressed in the following subsections.

5.1.1 Iteration modelling

During the initialisation, based on the inputs, the model determines whether an entity causes feedback. If that is the case, the entity is of the type gateway. The decision whether or not to feedback is made in the model based on the current state of the entity. The entity checks the total times the entity has been completed and verifies if that is below the set amount of iterations. If this is the case then the gateway entity sends a reset signal to the entities in the list of feedback accompanied with the ID of the sender (i.e. the current gateway entity). The entities receiving this reset signal stop their process if they are running. How this entity receiving the signal deals with this feedback is discussed in the next paragraph on rework modelling.

5.1.2 Rework modelling

Rework is modelled by sending reset signals sent between entities. If an entity receives a reset signal the entity is stopped and resets the progress of the activity to zero. Subsequently the entity forwards the signal to its succeeding activities to cause a trickle-down effect. This trickle-down effect accounts for the successive feed-forward rework as discussed by Cho and Eppinger [7]. This policy has been illustrated in Figure 6. As soon as the gateway entity triggers the feedback, all the work performed by completed activities 1, 2, 3 and 4 is reset. Also the work in progress in activity 5 is stopped and reset to zero. It is important to note that it is assumed that rework in a task always leads to rework in its succeeding tasks if the succeeding task has started or has already been completed before the reset signal.

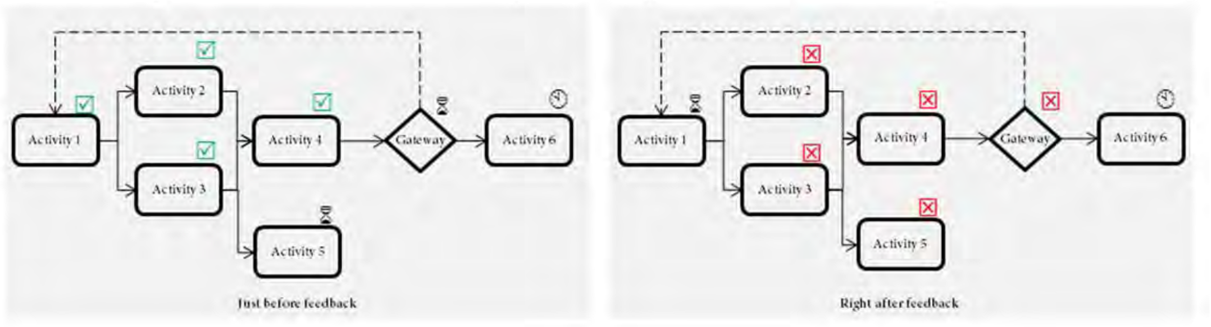


Figure 6: Illustration of rework policy

5.1.3 Collaboration modelling

Collaboration is modelled by using penalties for transactions between different resources. This penalty is a delay before starting the actual activity and is determined by the user. The penalty is not applicable to every activity. Each activity verifies if the resources of the previous tasks are a subset of its current resources, if this is not the case then the time to wait is accounted for. This applies in the case human resources are involved. In case of fully automated tasks, no delay is applied because of the assumption that the automated system has always a computer resource available.

5.2 Optimisation strategy

The simulator described in the previous section is able to analyse any given process architecture and output results important process KPIs like lead time and investment cost. The goal of this research is to provide insight in the trade-off between costs and benefits for the use of automation. Because of the multiple objective functions of interest, lead time and investment cost, a Multi-Objective Optimisation (MOO) problem is at hand. As illustrated in Figure 5, the process simulator is connected with an optimizer, with the final objective of finding the set of possible process architectures resulting in an optimal outcome for the multi-objective problem. These process architectures are those located on the previously discussed Pareto front of Figure 2.

In order to generate such a Pareto front, a possibility is to perform an exhaustive search; this would imply assessing all possible permutations of the design vector, hence all the possible combinations of level of automation for each activity in each task of the process. The total amount of possible permutation experiences a combinatorial explosion with an increasing number of activities. An exhaustive search would therefore result in too long computational time for the proof of concept purpose of this framework; hence a search algorithm is preferred.

Due to the discrete modelling of the levels of automation a gradient-based search method cannot be used. An evolutionary or genetic algorithm is proposed, specifically a Non-dominant Sorting Evolution algorithm. The algorithm from the OPTIMUS software application is used for implementation.

6 FRAMEWORK TECHNICAL IMPLEMENTATION AND FUNCTIONAL VERIFICATION

The elements discussed in previous paragraphs have been implemented in the integrated framework illustrated in Figure 7. This integrated framework assists in the knowledge acquisition, structuring, simulation and optimisation of the process architecture. In this case, the KE-chain tool was used for two purposes: as a systems integrator and Workflow Management System (WFS). It integrates the system by allowing all different modules and applications to exchange information. Furthermore it assists the user as a WFS by guiding through the different phases of the full methodology (see Figure 3).

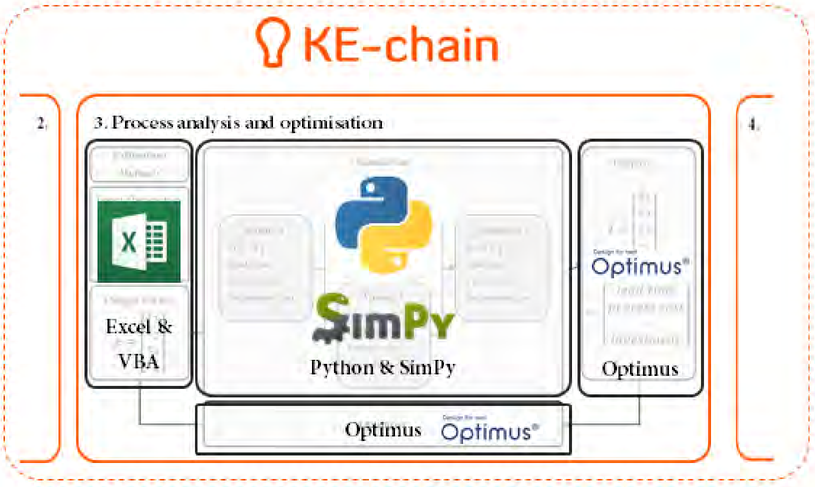


Figure 7: Integrated framework overview as part of the full methodology illustrated in Figure 3

In the first step the process is defined and its specific settings stated. To this purpose, Microsoft Excel in combination with custom Visual Basic for Application (VBA) scripts was used to provide a user friendly, and partly automated, interface to define the process to be analysed and optimized. In the second step the process is analysed and optimised using the simulation algorithm and optimisation strategy discussed earlier.

A number of simple test cases are discussed here to illustrate the functionality of the framework. A much more complex test case from an industrial application is discussed later in section 6.

The following test cases are all based on the same process configuration, referred to as *base case*. In each test case one characteristic is adjusted to demonstrate the difference in behaviour of the system. Flowcharts of the test cases are illustrated in Figure 8. All activities have an initial duration of 20 hours.

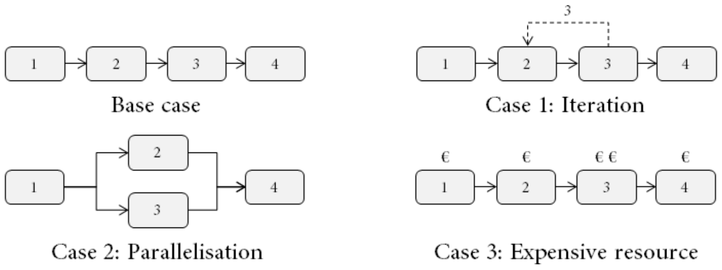


Figure 8: Flowcharts of multiple test cases

6.1.1 Test case 1: Iteration

This case displays the effect of iteration in a process. One iterative loop is added where task 3 feedbacks to task 2. In Figure 9 it can be seen that the Pareto front has shifted. Due to the iteration the process has a longer lead time. The slope of the Pareto front also has changed, implying that a larger lead time reduction than the base case can be obtained for a certain investment in automation solutions. This slope eventually matches with the slope of the base case when the iterative tasks have been fully automated.

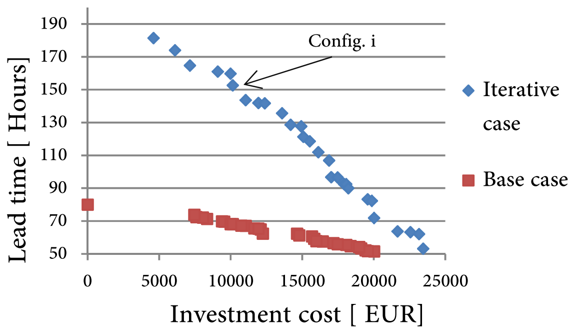


Figure 9: Pareto front for the lead time and investment cost for the base case and iterative case

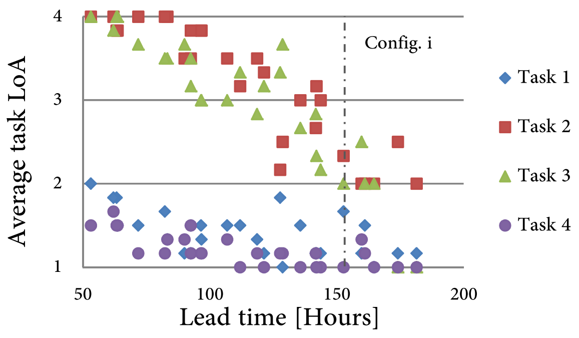


Figure 10: Average level of automation per task

Upon inspection of the levels of automation on the various activities in Figure 10 it can be seen that the points on the Pareto front correspond to process architectures with increasing levels of automation of the tasks involved in the iteration. These results match the expectations of the interviewed experts. In Figure 10 the points on 'vertical lines' correspond to a process architecture in Figure 9.

6.1.2 Test case 2: Parallelisation

In this case the precedence constraints on activities are changed and 2 and 3 can be processed in parallel. This case is investigated for two different scenarios since resource availability influences the results. In scenario I only one resource is available, scenario II has two resources. In Figure 11 it can be seen that the Pareto front of scenario I is similar to the base case. This is as expected since with only one resource the process is not able to process parallel activities.

For scenario II, however, a different front can be seen. The Pareto front flexes at a lead time of 30 hours. Upon investigation of the automation initiatives this can be explained. In Figure 12 the average levels of automation at different points on the Pareto of scenario II are plotted. Here it can be seen that at higher lead times (i.e. on the lower right side of the Pareto front) the non-parallel activities are first automated. Once these non-parallel activities have been (fully) automated the parallel activities (2 and 3) subsequently increases the level of automation. This is in accordance with the expectations since a higher level of automation for a parallel activity only becomes effective for a lower lead time if the other parallel task is also automated.

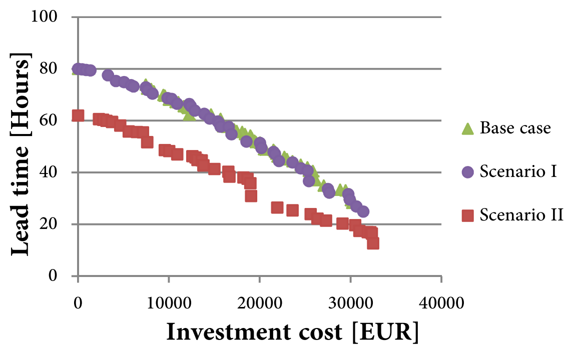


Figure 11: Pareto front for lead time and investment cost for parallelisation cases and base case

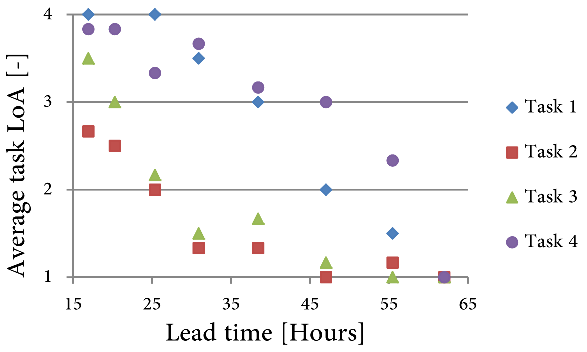


Figure 12: Selection of architectures on the Pareto front for Scenario II with average levels of automation

6.1.3 Test case 3: Resource cost

In this case one task utilises a different resource with a higher resource cost (in this fictitious case 100%). In this case it would be of no use to perform the multi-objective optimisation (MOO) for lead time and automation investment cost since resource cost has no effect on lead time and only a relative small effect on automation investment cost. Hence a MOO for lead time and investment cost would yield a similar Pareto front. Therefore a MOO for the lead time and the number of projects needed until Break Even Point was performed.

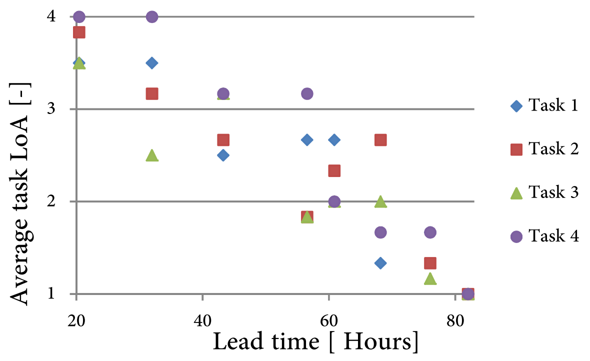


Figure 13: Average level of automation per task for base case for configuration on the Pareto front

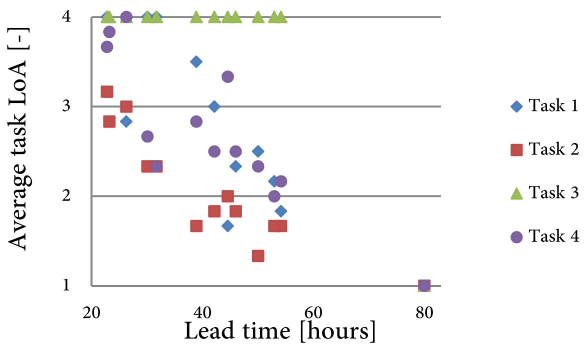


Figure 14: Average level of automation per task for test case 3 for configurations on the Pareto front

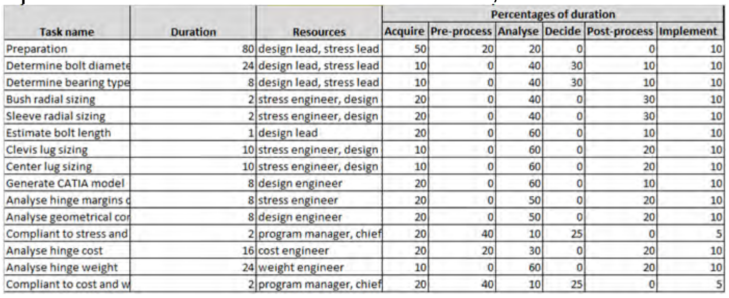
Figure 13 displays a selection of architectures on the Pareto front where it can be seen that all tasks are incrementally automated. When the cost of the resource of Task 3 is increased, a different graph is generated. This is illustrated in Figure 14, where it can be seen that Task 3 (using the expensive resource) is always fully automated. Hence, as expected, the optimization framework suggests automating first the task with high resource cost, ceteris paribus.

6.2 Industrial application case

The proposed methodology and optimization framework is applied to an industrial Product Development Process (PDP). The selected case is the conceptual design process of the hinge connections of a rudder assembly on the vertical tail of a business jet aircraft. This case provides a representative case of complex PDP, featuring many interesting characteristics: iterative loops resulting in rework, many involved departments and a mix of creative and repetitive tasks to name just a few.

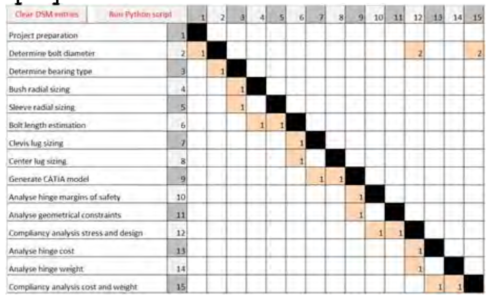
The process consists of 14 tasks for a total of 65 activities. The involved departments are stress engineering, design engineering, cost engineering and weight engineering. In total eight resources are involved, ranging from cost engineers to the project manager.

A summary of the inputs defining the process is provided in Figure 15 and 16. A full description of the input data is not relevant to this discussion, but can be found in [14].



Task name	Duration	Resources	Percentages of duration					
			Acquire	Pre-process	Analyse	Decide	Post-process	Implement
Preparation	80	design lead, stress lead	50	20	20	0	0	10
Determine bolt diamete	24	design lead, stress lead	10	0	40	30	10	10
Determine bearing type	8	design lead, stress lead	10	0	40	30	10	10
Bush radial sizing	2	stress engineer, design	20	0	40	0	30	10
Sleeve radial sizing	2	stress engineer, design	20	0	40	0	30	10
Estimate bolt length	1	design lead	20	0	60	0	10	10
Clevis lug sizing	10	stress engineer, design	10	0	60	0	20	10
Center lug sizing	10	stress engineer, design	10	0	60	0	20	10
Generate CATIA model	8	design engineer	20	0	60	0	10	10
Analyse hinge margins c	8	stress engineer	20	0	50	0	20	10
Analyse geometrical cor	8	design engineer	20	0	50	0	20	10
Compliant to stress and	2	program manager, chief	20	40	10	25	0	5
Analyse hinge cost	16	cost engineer	20	20	30	0	20	10
Analyse hinge weight	24	weight engineer	10	0	60	0	20	10
Compliant to cost and w	2	program manager, chief	20	40	10	25	0	5

Figure 15: High level task inputs of the case study



Clear DSM entries	Run Python script	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Project preparation	1	■														
Determine bolt diameter	2	1	■										2			2
Determine bearing type	3		1	■												
Bush radial sizing	4			1	■											
Sleeve radial sizing	5			1		■										
Bolt length estimation	6				1	1	■									
Clevis lug sizing	7						1	■								
Center lug sizing	8						1		■							
Generate CATIA model	9							1	1	■						
Analyse hinge margins of safety	10									1	■					
Analyse geometrical constraints	11									1		■				
Compliancy analysis stress and design	12										1	1	■			
Analyse hinge cost	13												1	■		
Analyse hinge weight	14												1		■	
Compliancy analysis cost and weight	15													1	1	■

Figure 16: Activity-based Design Structure Matrix of the case study

The simulation of the process architecture in the current state results in a lead time and total process time less than 5% different from the lead time and total process time estimated by the experts during interviews. While the estimation provided by the simulation framework is a bottom-up estimation, the one provided by the experts was top-down.

The result of the MOO can be seen in Figure 17, where a clear Pareto front is identified.

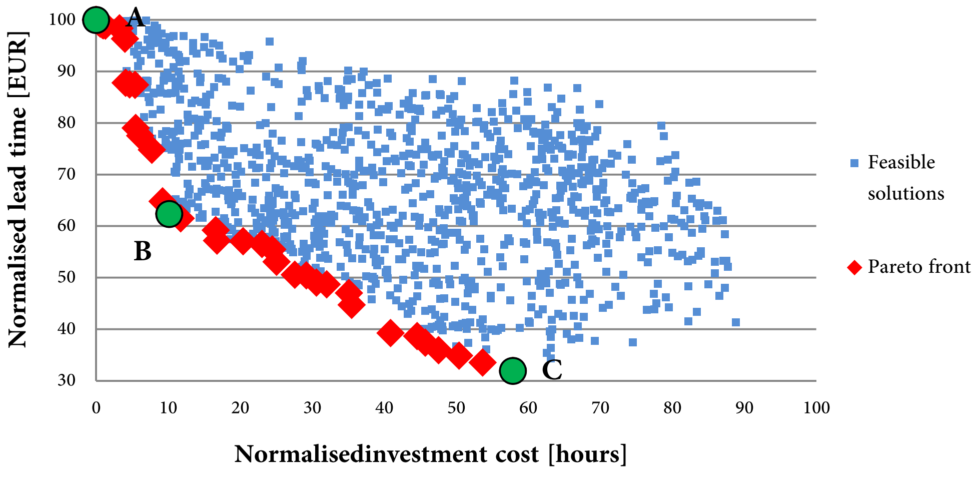


Figure 17: Pareto front for the Multi-Objective Optimisation for lead time and total investment cost

The outcome of three different process architectures on the Pareto front (indicated with A, B and C) is displayed in Table 5.

Table 5: Normalised overview of KPI's of selected architectures

	Architecture A	Architecture B	Architecture C
Lead time [Hours]	100	62,35	31,93
Process time {Hours}	100	49,75	22,27
Process cost [EUR]	100	51,45	23,28
Investment cost [EUR]	0	10,1	57,87

By investigating all the process architectures on the Pareto front some interesting findings can be done. Some tasks are not increased in their level of automation, whilst others are on a higher level of automation in most of the Pareto points. It can be seen that the iterative tasks are automated more frequently and the tasks outside the iterative loops (e.g. preparation) are only automated in the upper left region of the Pareto front (i.e. high investment cost, low lead time). Furthermore it can be observed that the ”Process” activities are the most frequently automated in the architectures on the Pareto front.

7 DISCUSSION

The results show that the proposed methodology is able to analyse a given process architecture and perform a multi-objective optimisation. The simulator is able to provide an estimation of the total lead time and investment cost for any process architecture, including different levels of automation on an activity level. A Pareto front trading off lead time and investment cost is generated by using a Non-dominant Sorting Evolution Algorithm.

The research has shown that the impact of automation can be estimated a priori, thereby offering the possibility to estimate the effect of automation in terms of lead time reduction and investment cost, on a given process architecture.

The methodology shows the potential for incremental innovation. It is able to simulate automation initiatives on different activities with different levels of automation. This methodology assists in determining what tasks and activities show the highest potential for this incremental innovation.

7.1 Limitations of the methodology

The authors are aware of some limitations of the presented methodology. Firstly, the model assumes deterministic activity durations, whilst in reality activity durations in the PDP are stochastic. Unfortunately, the computational cost would severely increase when accounting for stochastic effects.

This research uses deterministic rework modelling, hence a change always leads to rework in subsequent tasks. Rework probability, being the chance of a change leading to rework, is thus not taken into account. Furthermore it is assumed that rework has a constant duration equal to the initial duration, hence no learning effect is taken into account in the case studies. For this research it has been decided that the activities causing iterations and the number of times they cause iteration are predetermined in order to be able to have a deterministic model.

7.2 Model validity

Smith and Morrow [22] use the term ’face validity’ as a measure of validity of a PDP model. According to the described criteria this methodology would have high face validity. The proposed methodology and all underlying assumptions, theories and outcomes have been discussed with experts and according to their judgement the methodology is valid. Smith and Morrow define the next level of validation as the application of the methodology on existing but retrospective data in industry. The case study has shown that the methodology is applicable and can be used but due to insufficient retrospective data this was not validated completely.

7.3 Future research

The integrated framework has been developed in a modular way and can easily be extended and adjusted when needed. Process restructuring, as discussed in section 3, has not been formally implemented in the integrated framework. This topic will be addressed in research following this article.

Furthermore it is of great relevance to perform a sensitivity study on the input parameters. Based on this information the uncertainty of the generated Pareto front can be discussed.

Finally, time and cost coefficients for the proposed framework have been estimated using expert opinion, an inherently subjective approach. Quantitative approaches to determination of coefficients are currently being investigated.

REFERENCES

[1] Balogh, I., Ohlsson, K., Hansson, G. Å., Engström, T., & Skerfving, S. (2006). Increasing the degree of automation in a production system: consequences for the physical workload. *International Journal of Industrial Ergonomics*, *36*(4), 353-365.

[2] Brandao, R., & Wynn, M. (2009, February). Improving the New Product Development Process through ICT Systems in the Aerospace Industry–a Report on Case Study Research. In *Information, Process, and Knowledge Management, 2009. eKNOW'09. International Conference on* (pp. 147-152). IEEE.

[3] Brown, S. L., & Eisenhardt, K. M. (1995). Product development: Past research, present findings, and future directions. *Academy of* management *review*, *20*(2), 343-378.

[4] Browning, T. R., & Eppinger, S. D. (2002). Modeling impacts of process architecture on cost and schedule risk in product development. *Engineering Management, IEEE Transactions on*, *49*(4), 428-442.

[5] Browning, T. R. (1998). Use of Dependency Structure Matrices for Product Development Cycle Time Reduction. Paper presented at the Proceedings of the 5th *ISPE International Conference on Concurrent Engineering: Research and Applications (Japan),* Tokyo, July 15-17, pages 1–8.

[6] Browning, T. R., & Ramasesh, R. V. (2007). A survey of activity network-based process models for managing product development projects. *Production and Operations Management*, *16*(2), 217-240.

[7] Cho, S. H., & Eppinger, S. D. (2005). A simulation-based process model for managing complex design projects. *Engineering Management, IEEE Transactions on*, *52*(3), 316-328.

[8] Ha, S., & Suh, H. W. (2008). A timed colored Petri nets modeling for dynamic workflow in product development process. *Computers in industry*, *59*(2), 193-209.

[9] Hammer, M. (2001, March). Seven insights about processes. Paper presented at the *Proceedings of the Conference on Strategic Power Process Ensuring Survival Creating Competitive Advantage, Boston, MA, US*.

[10] Hart, J. J., & Valasek, J. (2010). *Methodology for prototyping increased levels of automation for spacecraft rendezvous functions*. Texas A&M University.

[11] Krishnan, V., & Ulrich, K. T. (2001). Product development decisions: A review of the literature. *Management science*, *47*(1), 1-21.

[12] Millson, M. R., Raj, S. P., & Wilemon, D. (1992). A survey of major approaches for accelerating new product development. *Journal of Product Innovation Management*, *9*(1), 53-69.

[13] Milton, N. R. (2007). Knowledge acquisition in practice: a step-by-step guide. London, England: Springer Science & Business Media.

[14] Mulder, B. (2015). A methodological approach for the optimisation of the product development process by the application of design automation. Unpublished MSc thesis, Delft University of Technology, Delft, Netherlands

[15] Muller, K., & Vignaux, T. (2003). Simpy: Simulating systems in python. *ONLamp. com Python Devcenter.*

[16] Reed, J. A., Follen, G. J., & Afjeh, A. A. (2000). Improving the aircraft design process using Web-based modeling and simulation. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, *10*(1), 58-83.

[17] Reinertsen, D. G. (2009). *The principles of product development flow: second generation lean product development* (Vol. 62). Redondo Beach, Canada: Celeritas.

[18] Sheridan, T. B., & Verplanck, W. L. (1978). Human and computer control of undersea teleoperators. Man-machine Systems Lab. Dept. of Mech. Eng.

[19] Schut, E. J., Kosman, S., & Curran, R. (2013). A Value Scan Methodology to Improve Industrial Operations. In *Concurrent Engineering Approaches for Sustainable Product Development in a Multi-Disciplinary Environment* (pp. 411-423). Springer London.

[20] Sheridan, T. B., & Parasuraman, R. (2005). Human-automation interaction.Reviews of human factors and ergonomics, 1(1), 89-129.

[21] Smith, R. P., & Eppinger, S. D. (1997). A predictive model of sequential iteration in engineering design. *Management Science*, *43*(8), 1104-1120.

[22] Smith, R. P., & Morrow, J. A. (1999). Product development process modeling.*Design studies*, *20*(3), 237-261.

[23] Terwiesch, C., Loch, C. H., & Meyer, A. D. (2002). Exchanging preliminary information in concurrent engineering: Alternative coordination strategies. *Organization Science*, *13*(4), 402-419.

[24] Van der Velden, C., Bil, C., & Xu, X. (2012). Adaptable methodology for automation application development. *Advanced Engineering Informatics*, *26*(2), 231-250.

[25] Verhagen, W. J., de Vrught, B., Schut, J., & Curran, R. (2015). A method for identification of automation potential through modelling of engineering processes and quantification of information waste. *Advanced Engineering Informatics*.

[26] Wickens, C. D., Li, H., Santamaria, A., Sebok, A., & Sarter, N. B. (2010, September). Stages and levels of automation: An integrated meta-analysis. Paper presented at the *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*(Vol. 54, No. 4, pp. 389-393). SAGE Publications.

[27] Wu, Z., Li, L., & Zhao, H. (2010, June). Simulation and analysis of schedule and cost of product development. Paper present at the *Proceedings of the Mechanic Automation and Control Engineering (MACE), 2010 International Conference on* (pp. 228-233). IEEE.

[28] Yassine, A., & Braha, D. (2003). Complex concurrent engineering and the design structure matrix method. *Concurrent Engineering*, *11*(3), 165-176.

B
PDP OPTIMISATION STEPS

The work presented in this thesis follows the work performed by Schut *et al.* [15] and Verhagen *et al.* [13] into development process optimisation. Both researches were conducted in close collaboration between the Faculty of Aerospace Engineering at the Delft University of Technology and the company KE-works. The research of Schut *et al.* resulted in a Value Scan for process improvements by means of reducing wasted time and optimising the information flow. This method was a very high level scan of the process (i.e. at low process granularity). In the succeeding research by Verhagen *et al.* the implementation of information flow automation in processes was assessed by using the IMPROVE method [13].
The steps as proposed in this appendix are based on these previous research projects and an analysis of current and past projects in which optimisation of the PDP is assessed by means of automation. As stated in Section 1.3.3, multiple different steps are involved in the optimisation of the PDP by the use of automation. In the process from start (e.g. request from a client) to the end (e.g. evaluation of the project) multiple steps were identified based on the previous research and analysis of three previous commercial projects by KE-works. These steps are illustrated in Figure B.1 and are discussed in more detail in the following paragraphs.

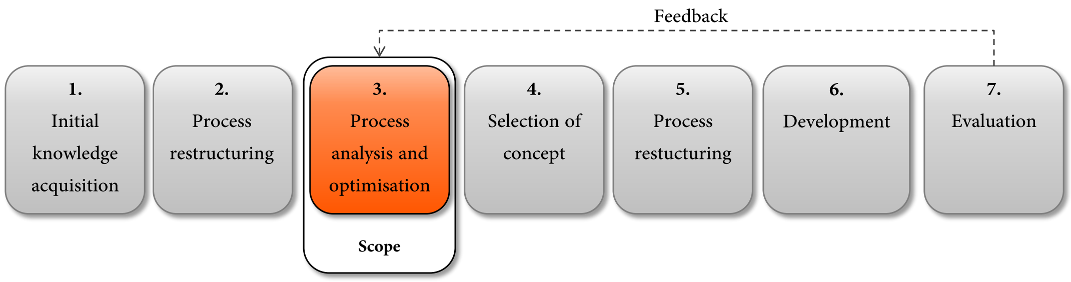


Figure B.1: Steps involved in methodology

B.1. STEP 1: INITIAL KNOWLEDGE ACQUISITION

At the start of a project an initial knowledge acquisition should be performed to gather the information about the process under consideration. This knowledge acquisition is focussed on *what, how, when* and *by whom* it happens. *What* is concerned with what tasks are executed in the process and what activities this task consists of. *How* is about the supporting tools and systems provided for the activities. *When* is about the timing of the task, when it is executed (e.g. what are the preceding tasks). *By whom* accounts for the resources used and the availability of resources within a project.

B.2. STEP 2: PROCESS RE-STRUCTURING

A process as elicitated during the knowledge acquisition often is not consistent and complete. In many cases it represents the way of working but the actual precedence constraints applicable to the project are incomplete. This step should complement the process and deliver a complete an consistent process (i.e. a process in which input and output relations comply with precedence constraints). Additionally the process could be improved by re-structuring the complete process and hence change the sequencing of tasks. Hereby adjusting the relations between the tasks.

B.3. Step 3: Process analysis and optimisation

In this step the process is analysed and optimised for the actual content of the activities in the tasks. The process is optimised for multiple objectives resulting in information to assist in making an informed decision on the optimal solution for the current problem under current constraints (e.g. budget, resources).

B.4. Step 4: Concept selection

The previous step generates information on optimal solutions. A decision should be made on the new process architecture taking into account all constraints applicable to the new process architecture. Also the expert judgement should be taken into account in this step since the optimisation might not account for certain aspects leading to significant alterations in for example lead time or investment cost.

B.5. Step 5: Re-structure the process architecture

Upon increasing the levels of automation in a process different activities might become fully automated. Restructuring of the process might be of use to cluster tasks based on their level of automation. Clustering the automated tasks to create large automated blocks might lead to lower overall lead time since no working hours need to be taken into account for fully automated chains of activities.

B.6. Step 6: Development and implementation

This is the step of actually developing and implementing the process architecture as determined in the previous steps. Also the aspects of testing and training are part of this step.

B.7. Step 7: Evaluation

Once the new process architecture has been implemented the new process architecture should be evaluated and benchmark tests should be performed. The information acquired in this evaluation should then feedback to the methods used during process analysis to update and improve the analysis and optimisation methods.

C

Non-dominating Sorting Genetic Algorithm

Three important aspects in NSGA-II are discussed in the following subsections. First the fast non-dominated sorting procedure, secondly then diversity preservation and finally the main loop of the algorithm. For a full discussion on the algorithm the author refers to [4].

C.1. Fast non-dominated sorting procedure

First for each solution (p) two entities are calculated: the domination count (n_p) and the set of solutions that the solution p dominates (S_p).
The solutions in the first non-dominated front have a domination count equal to zero ($n_p = 0$). For all solutions in the first non-dominated front each member (q) in the set S_p. Of each q the domination count is reduced by one, if the domination count of a member becomes zero it is stored in a separate list: Q. The members in Q belong to the second front. This procedure continues until all fronts have been identified. This sorting procedure is also shown in Algorithm 2.

C.2. Diversity preservation approach

As discussed in previous sections an evolutionary (or genetic) algorithm is able to deal with complex Pareto front shapes. It is desirable that the full shape (and design space) is discovered by the algorithm and therefore a sustainable diversity in the population is required.
The approach to obtain this diverse population in the NSGA-II is by means of a density estimation and a crowded-comparison operator.

Density estimation To get an estimate of the density of solutions around a specific solution a cuboid is generated around this solution. This has been illustrated for the solution i in Figure C.1. This cuboid is based on the distance from i to two Pareto solutions on either side in the directions of the objectives. The average side-length of this cuboid is defined as the crowding distance ($i_{distance}$).
The example in Figure C.1 displays the cuboid for two objective functions. The approach is however not limited to two objectives, it can handle any number of objectives. Therefore the population is sorted for an objective, and the boundary values are assigned as the infinite values. Using this infinite value the crowding distance is calculated with normalised values. This procedure is executed for all objectives. The normalised crowding distances per objective per solution are accumulated resulting in an overall crowding distance value.
This overall crowding distance value provides an indication of the relative proximity to other solutions and is used by the crowded-comparison operator discussed in the following paragraph.

Crowded-comparison operator To generate a uniformly spread-out Pareto front the crowded-comparison operator is applied. This operator guides the algorithm during the decision in the main loop if two equal solutions are present. Every individual solution i has the following two attributes: 1) a non-domination rank (i_{rank}) and 2) a crowding distance ($i_{distance}$).

Algorithm 2 Fast non-dominated sort [4]

```
1: for each $p \in P$ do
2:     $S_p = \emptyset$
3:     $n_p = 0$
4:     for each $q \in P$ do
5:         if $p \prec q$ then
6:             $S_p = S_p \cup q$
7:         else if $q \prec p$ then
8:             $n_p = n_p + 1$
9:         end if
10:    end for
11:    if $n_p = 0$ then
12:        $p_{rank} = 1$
13:        $F_1 = F_1 \cup p$
14:    end if
15: end for
16: $i = 1$
17: while $F_i \neq \emptyset$ do
18:     $Q = \emptyset$
19:     for each $p \in F_i$ do
20:         for each $q \in S_p$ do
21:             $n_q = n_q - 1$
22:             if $n_q = 0$ then
23:                 $q_{rank} = i + 1$
24:                 $Q = Q \cup q$
25:             end if
26:         end for
27:     end for
28:     $i = i + 1$
29:     $F_i = Q$
30: end while
```

The partial order $\prec_n$ is defined in Algorithm 3.

Algorithm 3 Crowded-comparison operator [4]

```
1: $i \prec_n j$
2: if $(i_{rank} < j_{rank})$
3: or$((i_{rank} = j_{rank})$ and $(i_{distance} > j_{distance}))$
```

Meaning that if two solutions are compared first the domination rank is checked. The solution with the lower (better) rank is selected. If the non-domination rank is equal the solution with the larger crowding distance (solution in less crowded region) is selected.

C.3. ALGORITHM MAIN LOOP

With the knowledge of the above described elements of the algorithm the main loop can be discussed. The optimisation starts with a random population, P_0. This initial population is sorted based on non-domination and a fitness value is assigned based on the non-domination level.
Based on tournament selection, recombination and mutation operators of the algorithm the offspring population (Q_0) of size N is created. With these two population (P_0 and Q_0) the following generations are created. The algorithm for the following generations is discussed in the following paragraphs.
For the following generations (t^{th}+1 generation) the algorithm as state in Algorithm 4 is used. First a combined population (R_t) is made based on P_t and Q_t, the size of R_t is thus $2N$. These solutions are now sorted using the non-dominated sort multiple sets of non-dominated fronts with F_1 being the best non-dominated set. The population of P_{t+1} is filled with the best solutions from the sorted fronts starting with solutions from F_1, subsequently F_2 and so forth until P_{t+1} consists of N solutions. Lets say that F_{last} is the last set to be added but that the total count of solutions in F_1 to F_{last} is larger than N. In this case the set F_{last} is sorted

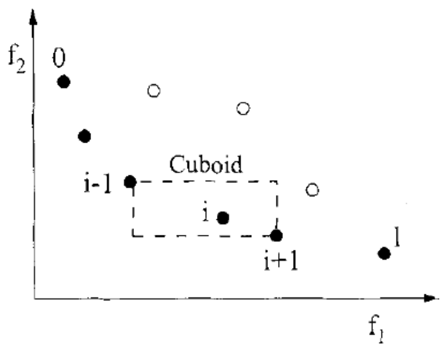


Figure C.1: Crowding distance calculation for solution i [4]

using the crowded-comparison operator.
The new population P_{t+1} is then used for tournament selection, recombination and mutation operators to generate the population of Q_{t+1}.

Algorithm 4 NSGA-II main loop [4]

```
1: $R_t = P_t \cup Q_t$
2: $F = fast-non-dominated-sort(R_t)$
3: $P_{t+1} = \emptyset$ and $i = 1$
4: while $|P_{t+1}| + |F_i| \geq N$ do
5:     $crowding-distance-assignment(F_i)$
6:     $P_{t+1} = P_{t+1} \cup F_i$
7:     $i = i + 1$
8: end while
9: Sort$(F_i, \prec_n)$
10: $P_{t+1} = P_{t+1} \cup F_i[1 : (N - |P_{t+1}|)]$
11: $Q_{t+1} = make-new-pop(P_{t+1})$
12: $t = t + 1$
```

D

Case description: Optimisation reliability

In Section 7.3 a case is described on a high level as illustrated in Figure 7.8. This case is used in the optimisation reliability analysis. This appendix provides the detailed information on the inputs as used in the simulation algorithm. Solely the resource and process parameters are discussed the simulation settings and methodology parameters are as discussed in Appendix G. Furthermore the default optimisation settings are provided in Table D.4.

D.1. Resource parameters

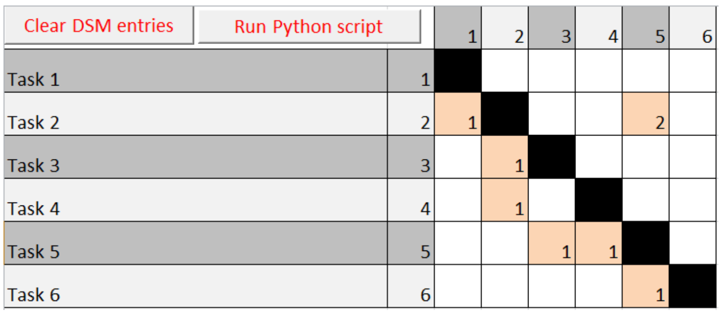
Information regarding the cost of the resources can be seen in Table D.1.

Table D.1: Definition of resources as initiated in the simulation

Resource type	Available amount	Hour rate
Dummy resource 1	1	50
Dummy resource 2	1	70

D.2. Process parameters

The parameters describing the process are provided in the following tables and figure. First in Table D.2 the parameters describing the process on a high level are provided. This information matches with the data structure of the input files for the simulator. Subsequently in Figure D.1 the Design Structure Matrix (DSM) on task level is provided. Next in Table D.3 the parameters for the process on an activity level are provided.



		1	2	3	4	5	6
Task 1	1	■					
Task 2	2	1	■			2	
Task 3	3		1	■			
Task 4	4		1		■		
Task 5	5			1	1	■	
Task 6	6					1	■

Figure D.1: Design Structure Matrix of the process defined on a task level

Table D.2: Parameters for the process description on a task level

			Percentages of duration					
Task name	**Duration**	**Resources**	**Acquire**	**Pre-process**	**Analyse**	**Decide**	**Post-process**	**Implement**
Task 1	10	dummy resource 1	10	20	30	0	20	20
Task 2	20	dummy resource 2	20	30	15	0	25	10
Task 3	30	dummy resource 1	10	20	20	20	20	10
Task 4	40	dummy resource 2	10	20	30	0	20	20
Task 5	30	dummy resource 1	20	30	15	0	25	10
Task 6	20	dummy resource 2	10	20	20	20	20	10

Table D.3: Parameters for the process description on an activity level

Name	**Duration**	**Current LoA**	**Max LoA**	**Resources**	**Type of activity**	**Development effort**	**Integration application**	**Integration class**	**Integration experience**
Acquire - Task 1	1	1	4	dummy resource 1	acquire	0		0	N/A
Pre-process - Task 1	2	1	4	dummy resource 1	process	0		0	N/A
Analyse - Task 1	3	1	4	dummy resource 1	analyse	10	FEM	3	1
Post-process - Task 1	2	1	4	dummy resource 1	process	0		0	N/A
Implement - Task 1	2	1	4	dummy resource 1	implement	0		0	N/A
Acquire - Task 2	4	1	4	dummy resource 2	acquire	0		0	N/A
Pre-process - Task 2	6	1	4	dummy resource 2	process	0		0	N/A
Analyse - Task 2	3	1	4	dummy resource 2	analyse	5	CATIA	2	1
Post-process - Task 2	5	1	4	dummy resource 2	process	0		0	N/A
Implement - Task 2	2	1	4	dummy resource 2	implement	1	ERP	1	1
Acquire - Task 3	3	1	4	dummy resource 1	acquire	0		0	N/A
Pre-process - Task 3	6	1	4	dummy resource 1	process	0		0	N/A
Analyse - Task 3	6	1	4	dummy resource 1	analyse	0		0	N/A
Decide - Task 3	6	1	4	dummy resource 1	decide	1	CRM	0	1
Post-process - Task 3	6	1	4	dummy resource 1	process	0		0	N/A
Implement - Task 3	3	1	4	dummy resource 1	implement	0		0	N/A

Continued on next page

Table D.3 – *Continued from previous page*

Name	**Duration**	**Current LoA**	**Max LoA**	**Resources**	**Type of activity**	**Development effort**	**Integration application**	**Integration class**	**Integration experience**
Acquire - Task 4	4	1	4	dummy resource 2	acquire	0		0	N/A
Pre-process - Task 4	8	1	4	dummy resource 2	process	0		0	N/A
Analyse - Task 4	12	1	4	dummy resource 2	analyse	5	ERP	1	1
Post-process - Task 4	8	1	4	dummy resource 2	process	0		0	N/A
Implement - Task 4	8	1	4	dummy resource 2	implement	0		0	N/A
Acquire - Task 5	6	1	4	dummy resource 1	acquire	0		0	N/A
Pre-process - Task 5	9	1	4	dummy resource 1	process	0		0	N/A
Analyse - Task 5	4,5	1	4	dummy resource 1	analyse	6	CATIA	2	1
Post-process - Task 5	7,5	1	4	dummy resource 1	process	0		0	N/A
Implement - Task 5	3	1	4	dummy resource 1	implement	0		0	N/A
Acquire - Task 6	2	1	4	dummy resource 2	acquire	0		0	N/A
Pre-process - Task 6	4	1	4	dummy resource 2	process	0		0	N/A
Analyse - Task 6	4	1	4	dummy resource 2	analyse	2	CRM	1	1
Decide - Task 6	4	1	4	dummy resource 2	decide	0		0	N/A
Post-process - Task 6	4	1	4	dummy resource 2	process	0		0	N/A
Implement - Task 6	2	1	4	dummy resource 2	implement	0		0	N/A

D.3. OPTIMISATION SETTINGS

Table D.4: Default parameter settings as used in the optimisation reliability analysis

Optimisation parameter	**Value**
Target front size	25
Population size	25
Start population	No method selected
First population size	25
Random seed	52
Weighting factor	0,7
Inverse crossover probability	0,85
Maximum number of iterations	50

E

Optimisation results: Robustness analysis

In Section 7.3.3 the robustness of the optimisation algorithm is discussed. Therefore multiple optimisations are performed using different settings. In this appendix the results of two optimisations using identical optimisation settings are displayed.
The optimisation results in different Pareto optimal solutions for that specific optimisation. Two profiles with pareto optimal solutions and corresponding design vectors are shown in Figure E.1 and Figure E.2. The optimisation settings for the two optimisations are identical, except for the random seed, and can be found in Table E.1.
In Figures E.1 and E.2 the design vectors for Pareto optimal solutions are plotted per row. Each column is an activity in the process. Hence the figure provides an overview of the different process architectures on the Pareto front. The rows are sorted on lead time (highest lead time on top).

Table E.1: Optimisation settings to verify the optimisation robustness as discussed in Section 7.3.3

Optimisation parameter	Value
Target front size	60
Population size	60
Start population	No method selected
First population size	60
Random seed	41 (E.1) and 39 (E.2)
Weighting factor	0,7
Inverse crossover probability	0,85
Maximum number of iterations	50

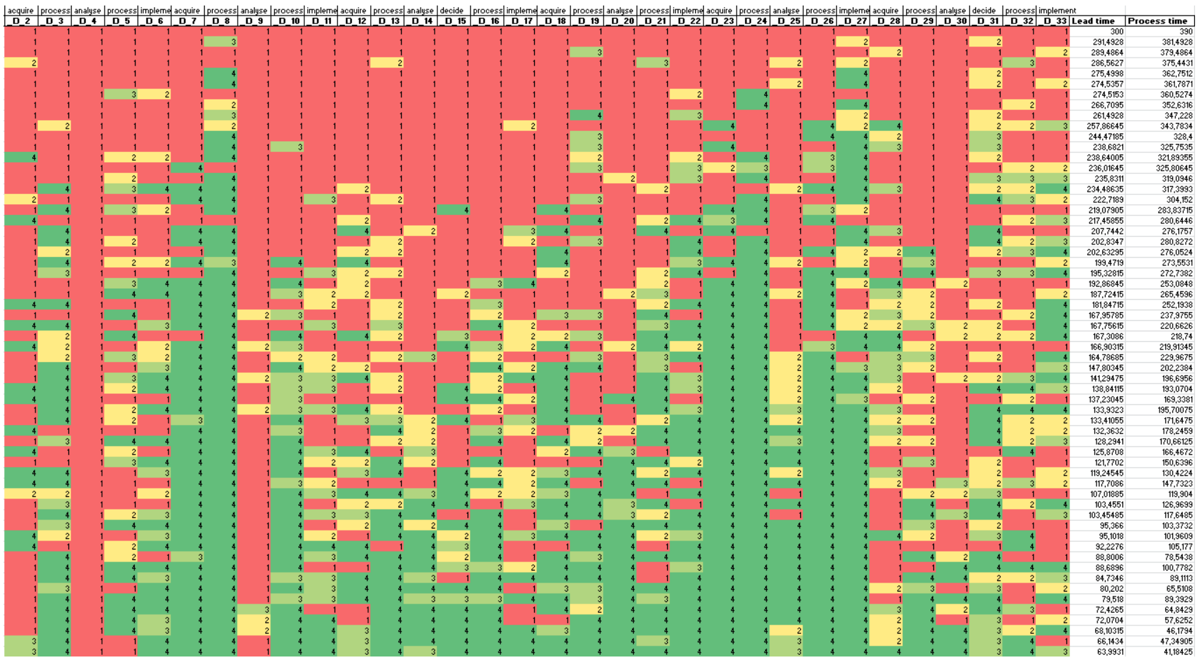


Figure E.1: Visually enhanced overview of all Pareto optimal design vectors for first run

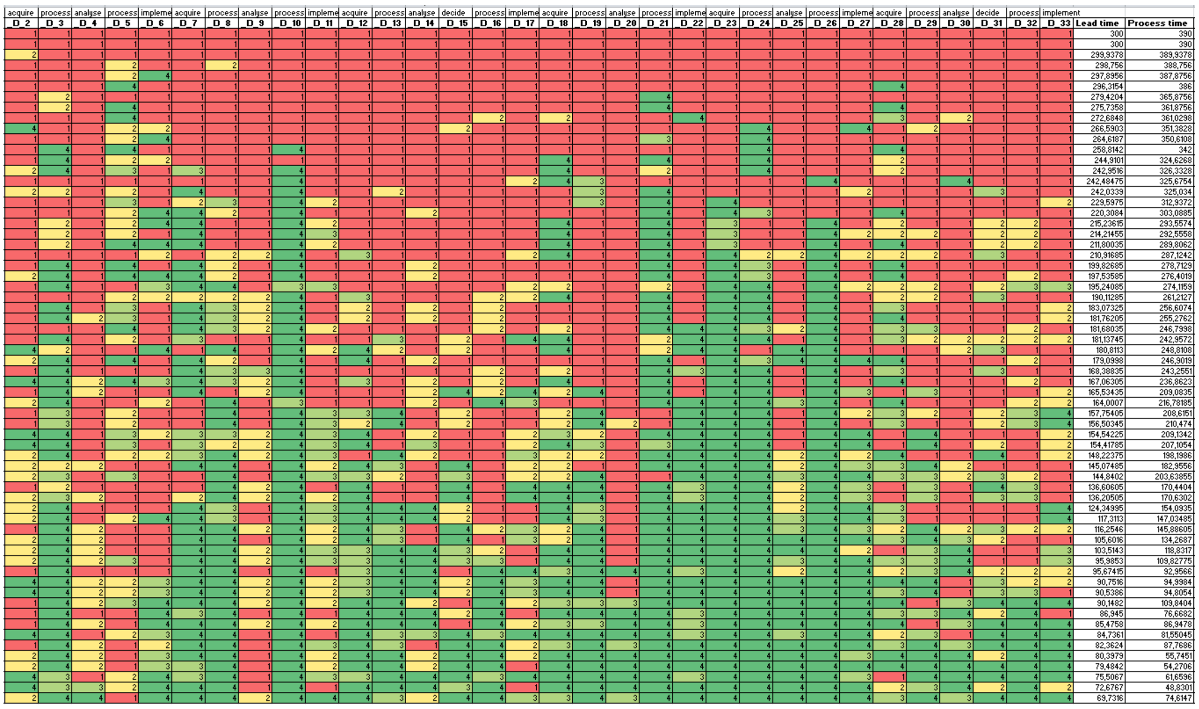


Figure E.2: Visually enhanced overview of all Pareto optimal design vectors for second run

F
Case description: Optimisation convergence

In Section 7.3.4 the convergence of the optimisation to an actual optimal solution is discussed. Therefore a specific case is used. Specifications of this case are described in this appendix. First on a high level the flowchart is presented, followed by more detailed inputs as provided to the simulation and optimisation modules.

F.1. Process flowchart

In Figure F.1 a flowchart of the process on task level is presented.

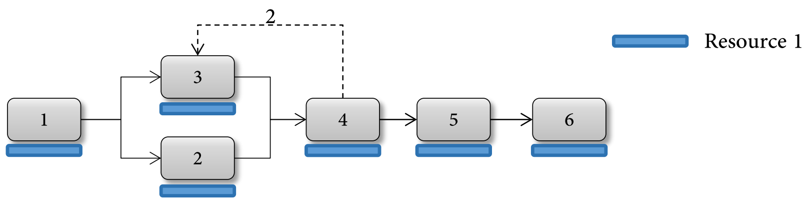


Figure F.1: Workflow of the process on a task level with indicated resources

F.2. Resource parameters

Information regarding the cost of the resources can be seen in Table F.1.

Table F.1: Definition of resources as initiated in the simulation

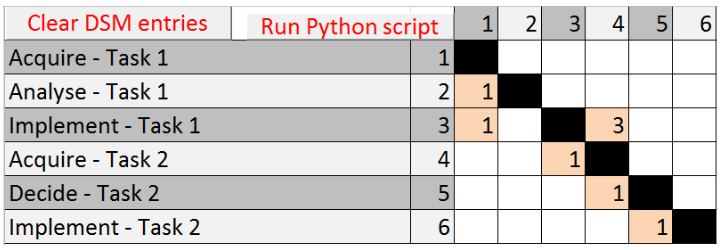
Resource type	**Available amount**	**Hour rate**
Resource 1	2	100

F.3. Process parameters

The parameters describing the process are provided in the following tables and figure. First in Table F.2 the parameters describing the process on a high level are provided. This information matches with the data structure of the input files for the simulator. Subsequently in Figure F.2 the DSM on task level is provided. Next in Table F.3 the parameters for the process on an activity level are provided.

Table F.2: Parameters for the process description on a task level

			Percentages of duration					
Task name	**Duration**	**Resources**	**Acquire**	**Pre-process**	**Analyse**	**Decide**	**Post-process**	**Implement**
Task 1	20	resource 1	30	0	50	0	0	20
Task 2	20	resource 1	20	0	0	50	0	30
Task 3	20	resource 1	10	0	80	0	0	10
Task 4	20	resource 1	30	0	50	0	0	20
Task 5	20	resource 1	20	0	0	50	0	30
Task 6	20	resource 1	10	0	80	0	0	10



Clear DSM entries	Run Python script	1	2	3	4	5	6
Acquire - Task 1	1	■					
Analyse - Task 1	2	1	■				
Implement - Task 1	3	1		■	3		
Acquire - Task 2	4			1	■		
Decide - Task 2	5				1	■	
Implement - Task 2	6					1	■

Figure F.2: Design Structure Matrix of the process defined on a task level

Table F.3: Parameters for the process description on an activity level

Name	**Duration**	**Current LoA**	**Max LoA**	**Resources**	**Type of activity**	**Development effort**	**Integration application**	**Integration class**	**Integration experience**
Acquire - Task 1	6	4	4	resource 1	acquire	0		0	N/A
Analyse - Task 1	10	3	4	resource 1	analyse	5	ERP	0	1
Implement - Task 1	4	3	4	resource 1	implement	0		0	N/A
Acquire - Task 2	4	4	4	resource 1	acquire	0		0	N/A
Decide - Task 2	10	3	4	resource 1	decide	3	ERP	0	1
Implement - Task 2	6	4	4	resource 1	implement	0		0	N/A
Acquire - Task 3	2	3	4	resource 1	acquire	0		0	N/A
Analyse - Task 3	16	3	4	resource 1	analyse	1	CATIA	0	1
Implement - Task 3	2	3	4	resource 1	implement	0		0	N/A
Acquire - Task 4	6	4	4	resource 1	acquire	0		0	N/A
Analyse - Task 4	10	3	4	resource 1	analyse	6	ERP	0	1
Implement - Task 4	4	3	4	resource 1	implement	0		0	N/A
Acquire - Task 5	4	3	4	resource 1	acquire	0		0	N/A
Decide - Task 5	10	4	4	resource 1	decide	2	CATIA	0	1
Implement - Task 5	6	3	4	resource 1	implement	0		0	N/A
Acquire - Task 6	2	4	4	resource 1	acquire	0		0	N/A
Analyse - Task 6	16	3	4	resource 1	analyse	0		0	N/A
Implement - Task 6	2	4	4	resource 1	implement	0		0	N/A

F.4. OPTIMISATION SETTINGS

Table F.4: Default parameter settings as used in the optimisation convergence analysis

Optimisation parameter	Value
Target front size	50
Population size	50
Start population	No method selected
First population size	20
Random seed	52
Weighting factor	0,7
Inverse crossover probability	0,85
Maximum number of iterations	50

G
Case description: Hinge connection

This appendix includes an overview of the used input parameters by the simulation program. Due to confidentially some values are normalised or redacted and replaced with an indicative value (hence in the order of magnitude). Normalised values and indicative values are marked with a $\star$ and $\diamond$ respectively.

G.1. Simulation settings

The parameters in Table G.1 are the input parameters used for the settings of the simulation.

Table G.1: Simulation settings input parameters used in case study (redacted)

Name	Value	Explanation
Iteration behaviour	1	Determines the behaviour in case of an infinite loop. (1=simulation stops if infinite loop is detected, 2= if infinite loop is detected the gateway under considerations becomes a regular task, 3= iteration counter is lowered each time. More information is available in guide.
Maximum task runs	50	Maximum number of times a single task can run (completed) before the simulation is assumed to be in an infinite loop.
multirun	False	True or False
Maximum level of automation	4	Maximum level of automation as defined by the user
Server	False	Availability of a shared server within the project in current sate
Server cost	$3000^{\diamond}$	Cost of license for server with central storage capabilities in euros
License	False	Availability of a KE-chain license within the project in the current state
License cost	$4000^{\diamond}$	Cost of license for workflow management system in euros
Project re-use	1	Number of project to discount the total invest over
Time unit	hours	The time unit of the duration item in the "task definition" sheet
Hour rate knowledge engineer	$125^{\diamond}$	Hour rate of a knowledge engineer as charged to client
Hour rate developer	$100^{\diamond}$	Hour rate of a developer as charged to client
Initial process cost	$100^{\star}$	Process cost of the as-is situation of the process
Python full path name	C:\Users\Mulder \Graduation \Code	e.g. "C:\Users\John.Doe\Files\Code"

G.2. Methodology parameters

This methodology is supported by multiple parameters and matrices. These are listed in the following tables. First in Table G.2 various parameters used in cost time calculation are listed. Secondly Table G.3 lists the Duration Matrix and Cost Matrices. Thirdly the cost of integration matrix can be seen in Table G.4.

Table G.2: Methodology input parameters used in case study (redacted)

Parameter	**Value**
$k_{KA_{int}}$	$2^{\diamond}$
$k_{KA_{ext}}$	$0{,}2^{\diamond}$
k_{dev}	$3^{\diamond}$
k_{conf}	$1^{\diamond}$
k_{mgt}	$1{,}1^{\diamond}$
$k_{training}$	$1{,}07^{\diamond}$
$t_{tostart}$	$0{,}5^{\diamond}$

Continued on next page

Table G.3: Methodology matrix input parameters used in case study (redacted)

Time$^{\diamond}$				
acquire	1	0.80	0,50	0,10
process	1	0.70	0.40	0.20
analyse	1	0.90	0.60	0.15
decide	1	0.65	0.40	0.30
implement	1	0.60	0.30	0.05

Knowledge acquisition$^{\diamond}$				
acquire	0	0.33	0.66	1
process	0	0.33	0.66	1
analyse	0	0.33	0.66	1
decide	0	0.33	0.66	1
implement	0	0.33	0.66	1

Development$^{\diamond}$				
acquire	0	0	0	1
process	0	0.1	0.6	1
analyse	0	0.4	0,9	1
decide	0	0	0,7	1
implement	0	0	0.2	1

Integration$^{\diamond}$				
acquire	0	0	0	1
process	0	0	0	1
analyse	0	0	0	1
decide	0	0	0	1
implement	0	0	0	1

Configuration$^{\diamond}$				
acquire	0	0,3	0,7	1
process	0	0,3	0,7	1
analyse	0	0,3	0,9	1
decide	0	0,3	0,7	1
implement	0	0,3	0,7	1

License$^{\diamond}$				
acquire	0	0	0	1
process	0	0	0	1
analyse	0	0	0	1
decide	0	0	0	1
implement	0	0	0	1

Server$^{\diamond}$				
acquire	0	0	1	0
process	0	0	1	0
analyse	0	0	1	0
decide	0	0	1	0
implement	0	0	1	0

Table G.4: Methodology input parameters for the cost of integration in case study

	Class 0	**Class 1**	**Class 2**	**Class 3**
Integration cost (IC) [€]	0	4500	11000	35000

G.3. RESOURCE PARAMETERS

Information regarding the cost of the resources can be seen in Table G.5.

Table G.5: Definition of resources as initiated in the simulation

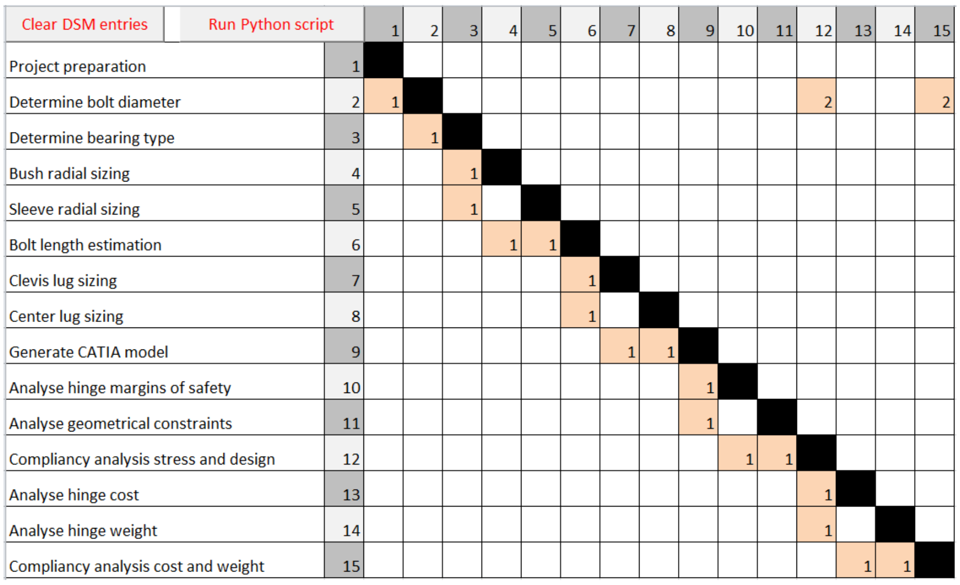
Resource type	**Available amount**	**Hour rate**
design lead	1	$90^\star$
stress lead	1	$90^\star$
stress engineer	1	$80^\star$
design engineer	1	$80^\star$
program manager	1	$100^\star$
chief engineer	1	$100^\star$
weight engineer	1	$80^\star$
cost engineer	1	$80^\star$

G.4. PROCESS PARAMETERS

The parameters describing the process are provided in the following tables and figure. First in Table G.6 the parameters describing the process on a high level are provided. This information matches with the data structure of the input files for the simulator. Subsequently in Figure G.1 the DSM on task level is provided. Next in Table G.7 the parameters for the process on an activity level are provided.

Table G.6: Parameters for the process description on a task level

			Percentages of duration					
Task name	**Duration**	**Resources**	**Acquire**	**Pre-process**	**Analyse**	**Decide**	**Post-process**	**Implement**
Preparation	100⋆	design lead, stress lead	50	20	20	0	0	10
Determine bolt diameter	30⋆	design lead, stress lead	10	0	40	30	10	10
Determine bearing type	10⋆	design lead, stress lead	10	0	40	30	10	10
Bush radial sizing	2,5⋆	stress engineer, design engineer	20	0	40	0	30	10
Sleeve radial sizing	2,5⋆	stress engineer, design engineer	20	0	40	0	30	10
Estimate bolt length	1,3⋆	design lead	20	0	60	0	10	10
Clevis lug sizing	10⋆	stress engineer, design engineer	10	0	60	0	20	10
Center lug sizing	10⋆	stress engineer, design engineer	10	0	60	0	20	10
Generate CATIA model	10⋆	design engineer	20	0	60	0	10	10
Analyse hinge margins of safety	10⋆	stress engineer	20	0	50	0	20	10
Analyse geometrical constraints	10⋆	design engineer	20	0	50	0	20	10
Compliant to stress and design?	2,5⋆	program manager, chief engineer	20	40	10	25	0	5
Analyse hinge cost	20⋆	cost engineer	20	20	30	0	20	10
Analyse hinge weight	30⋆	weight engineer	10	0	60	0	20	10
Compliant to cost and weight?	2,5⋆	program manager, chief engineer	20	40	10	25	0	5



Clear DSM entries	Run Python script	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Project preparation	1	■														
Determine bolt diameter	2	1	■										2			2
Determine bearing type	3		1	■												
Bush radial sizing	4			1	■											
Sleeve radial sizing	5			1		■										
Bolt length estimation	6				1	1	■									
Clevis lug sizing	7						1	■								
Center lug sizing	8						1		■							
Generate CATIA model	9							1	1	■						
Analyse hinge margins of safety	10									1	■					
Analyse geometrical constraints	11									1		■				
Compliancy analysis stress and design	12										1	1	■			
Analyse hinge cost	13												1	■		
Analyse hinge weight	14												1		■	
Compliancy analysis cost and weight	15													1	1	■

Figure G.1: Design Structure Matrix of the process defined on a task level

Table G.7: Parameters for the process description on an activity level

Name	**Duration***	**Current LoA**	**Max LoA**	**Resources**	**Type of activity**	**Development effort***	**Integration application**	**Integration class**	**Integration experience**
Acquire - Preparation	100	1	4	design lead, stress lead	acquire	100		0	N/A
Pre-process - Preparation	40	1	4	design lead, stress lead	process	33		0	N/A
Analyse - Preparation	40	1	4	design lead, stress lead	analyse	33		0	N/A
Implement - Preparation	20	1	4	design lead, stress lead	implement	0		0	N/A
Acquire - Determine bolt diameter	6	1	4	design lead, stress lead	acquire	0		0	N/A
Analyse - Determine bolt diameter	24	2	4	design lead, stress lead	analyse	12		0	N/A
Decide - Determine bolt diameter	18	1	3	design lead, stress lead	decide	0		0	N/A
Post-process - Determine bolt diameter	6	1	4	design lead, stress lead	process	0		0	N/A
Implement - Determine bolt diameter	6	1	4	design lead, stress lead	implement	0		0	N/A
Acquire - Determine bearing type	2	1	4	design lead, stress lead	acquire	0		0	N/A

Continued on next page

Table G.7 – *Continued from previous page*

Name	Duration⋆	Current LoA	Max LoA	Resources	Type of activity	Development effort⋆	Integration application	Integration class	Integration experience
Analyse - Determine bearing type	8	1	4	design lead, stress lead	analyse	12		0	N/A
Decide - Determine bearing type	6	1	4	design lead, stress lead	decide	0		0	N/A
Post-process - Determine bearing type	2	1	4	design lead, stress lead	process	0		0	N/A
Implement - Determine bearing type	2	1	4	design lead, stress lead	implement	0		0	N/A
Acquire - Bush radial sizing	1	1	4	stress engineer, design engineer	acquire	0		0	N/A
Analyse - Bush radial sizing	2	1	4	stress engineer, design engineer	analyse	6		0	N/A
Post-process - Bush radial sizing	1,5	1	4	stress engineer, design engineer	process	2,5		0	N/A
Implement - Bush radial sizing	0.5	1	4	stress engineer, design engineer	implement	0		0	N/A
Acquire - Sleeve radial sizing	1	1	4	stress engineer, design engineer	acquire	0		0	N/A
Analyse - Sleeve radial sizing	2	1	4	stress engineer, design engineer	analyse	6		0	N/A
Post-process - Sleeve radial sizing	1,5	1	4	stress engineer, design engineer	process	2,5		0	N/A
Implement - Sleeve radial sizing	1	1	4	stress engineer, design engineer	implement	0		0	N/A
Acquire - Estimate bolt length	1	1	4	design lead	acquire	0		0	N/A
Analyse - Estimate bolt length	1.5	1	4	design lead	analyse	6		0	N/A
Post-process - Estimate bolt length	0,25	1	4	design lead	process	2,5		0	N/A
Implement - Estimate bolt length	0,25	1	4	design lead	implement	0		0	N/A

Continued on next page

Table G.7 – *Continued from previous page*

Name	**Duration**⋆	**Current LoA**	**Max LoA**	**Resources**	**Type of activity**	**Development effort**⋆	**Integration application**	**Integration class**	**Integration experience**
Acquire - Clevis lug sizing	2	1	4	stress engineer, design engineer	acquire	0		0	N/A
Analyse - Clevis lug sizing	12	3	4	stress engineer, design engineer	analyse	6	TH3 tools	1	1
Post-process - Clevis lug sizing	4	1	4	stress engineer, design engineer	process	2,5		0	N/A
Implement - Clevis lug sizing	2	1	4	stress engineer, design engineer	implement	0		0	N/A
Acquire - Center lug sizing	2	1	4	stress engineer, design engineer	acquire	0		0	N/A
Analyse - Center lug sizing	12	3	4	stress engineer, design engineer	analyse	6	TH3 tools	1	1
Post-process - Center lug sizing	4	1	4	stress engineer, design engineer	process	1,5		0	N/A
Implement - Center lug sizing	2	1	4	stress engineer, design engineer	implement	0		0	N/A
Acquire - Generate CATIA model	4	1	4	design engineer	acquire	0		0	N/A
Analyse - Generate CATIA model	12	2	4	design engineer	analyse	100	CATIA	2	0,8
Post-process - Generate CATIA model	2	2	4	design engineer	process	3		0	N/A
Implement - Generate CATIA model	2	1	4	design engineer	implement	0		0	N/A
Acquire - Analyse hinge margins of safety	2	1	4	stress engineer	acquire	0		0	N/A
Analyse - Analyse hinge margins of safety	10	2	4	stress engineer	analyse	17	TH3 tools	1	1
Post-process - Analyse hinge margins of safety	2	2	4	stress engineer	process	1,5		0	N/A
Implement - Analyse hinge margins of safety	2	1	4	stress engineer	implement	0		0	N/A

Continued on next page

Table G.7 – *Continued from previous page*

Name	**Duration***	**Current LoA**	**Max LoA**	**Resources**	**Type of activity**	**Development effort***	**Integration application**	**Integration class**	**Integration experience**
Acquire - Analyse geometrical constraints	4	1	4	design engineer	acquire	0		0	N/A
Analyse - Analyse geometrical constraints	10	1	4	design engineer	analyse	30	CATIA	2	0,8
Post-process - Analyse geometrical constraints	4	2	4	design engineer	process	1		0	N/A
Implement - Analyse geometrical constraints	2	1	4	design engineer	implement	0		0	N/A
Acquire - Compliant to stress and design?	1	1	4	program manager, chief engineer	acquire	0		0	N/A
Pre-process - Compliant to stress and design?	2	1	4	program manager, chief engineer	process	1,5		0	N/A
Analyse - Compliant to stress and design?	0,5	1	4	program manager, chief engineer	analyse	13		0	N/A
Decide - Compliant to stress and design?	1,25	1	3	program manager, chief engineer	decide	0		0	N/A
Implement - Compliant to stress and design?	0,25	1	4	program manager, chief engineer	implement	0		0	N/A
Acquire - Analyse hinge cost	6	1	4	cost engineer	acquire	0		0	N/A
Pre-process - Analyse hinge cost	6	2	4	cost engineer	process	9		0	N/A
Analyse - Analyse hinge cost	12	3	4	cost engineer	analyse	25	Excel	1	0,5
Post-process - Analyse hinge cost	8	3	4	cost engineer	process	3		0	N/A
Implement - Analyse hinge cost	4	1	4	cost engineer	implement	0		0	N/A
Acquire - Analyse hinge weight	6	1	4	weight engineer	acquire	0		0	N/A
Analyse - Analyse hinge weight	36	2	4	weight engineer	analyse	33	CATIA	2	0,8
Post-process - Analyse hinge weight	12	1	4	weight engineer	process	3		0	N/A
Implement - Analyse hinge weight	6	1	4	weight engineer	implement	0		0	N/A
Acquire - Compliant to cost and weight?	1	1	4	program manager, chief engineer	acquire	0		0	N/A

Continued on next page

Table G.7 – *Continued from previous page*

Name	**Duration**⋆	**Current LoA**	**Max LoA**	**Resources**	**Type of activity**	**Development effort**⋆	**Integration application**	**Integration class**	**Integration experience**
Pre-process - Compliant to cost and weight?	2	1	4	program manager, chief engineer	process	1,5		0	N/A
Analyse - Compliant to cost and weight?	0,5	1	4	program manager, chief engineer	analyse	13		0	N/A
Decide - Compliant to cost and weight?	1,25	1	3	program manager, chief engineer	decide	0		0	N/A
Implement - Compliant to cost and weight?	0,25	1	4	program manager, chief engineer	implement	0		0	N/A

H
Optimisation results: Hinge connection

The figures below show the optimisation profiles for the optimisations of the case study. First in Figure H.1 a summary is given of the levels of automations per task for optimisation one. Next in Figure H.2 the profiles of levels of automation for optimisation 1 are plotted on a task level for all Pareto optimal solutions. Here the average level of automation of a task is used. This is determined by taking the average of the different activities in the task. Next in Figure H.3 a similar profile is displayed but on an activity level.
Next in Figures H.5 and H.6 similar profiles are plotted for optimisation 2.

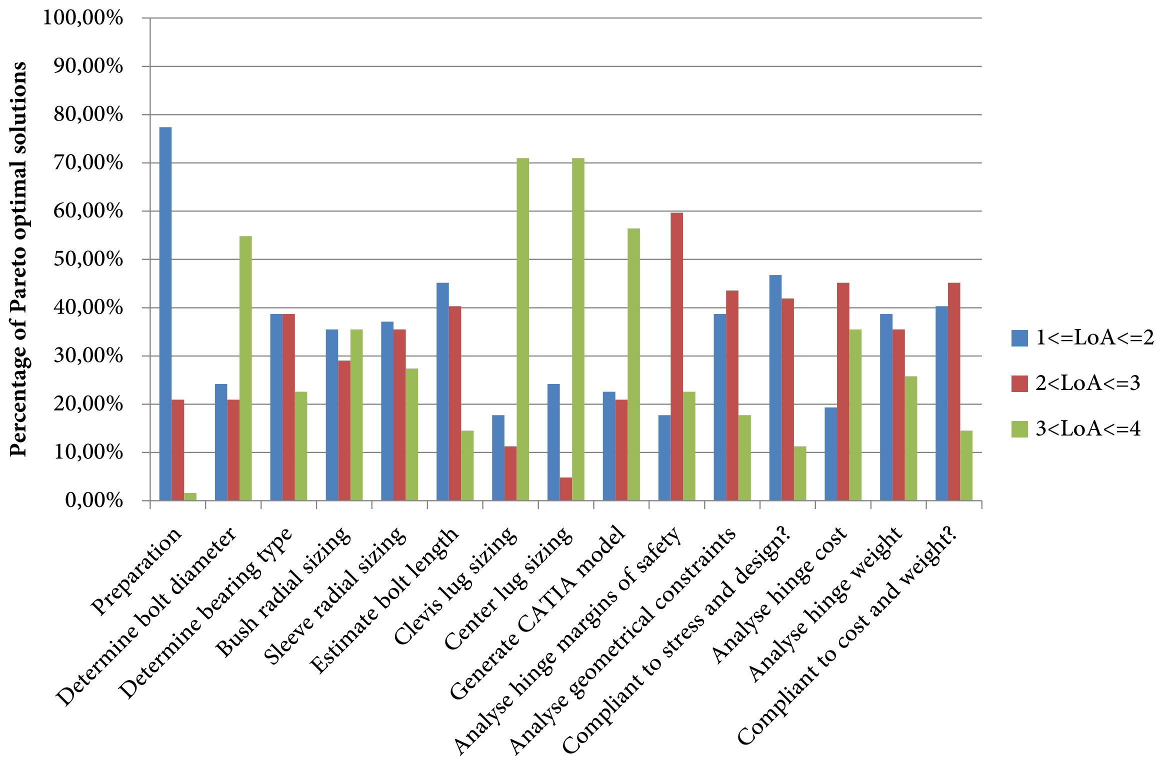


Figure H.1: Summary of the different levels of automation per task for optimisation 1

Preparation	Determine bolt diameter	Determine bearing type	Bush radial sizing	Sleeve radial sizing	Estimate bolt length	Clevis lug sizing	Center lug sizing	Generate CATIA model	Analyse hinge margins of safety	Analyse geometrical constraints	Compliant to stress and design?	Analyse hinge cost	Analyse hinge weight	Compliant to cost and weight?	Normalised lead time
1	1,2	1	1	1	1	1,5	1,5	1,5	1,5	1,25	1	2	1,25	1	100
1	1,2	1,2	1,25	1	1,25	1,5	1,5	1,5	1,5	1,25	1	2	1,25	1	100
1	1,2	1,2	1	1	1	1,75	1,75	1,5	1,5	1,5	1	2	1,25	1	100
1	1,2	1	1	1	1	1,5	1,5	1,5	1,75	1,25	1	2	1,25	1	100
1,25	1,4	1,2	1	1,25	1	1,75	1,5	1,75	2	1,5	1	2	1,5	1,2	99
1	1,2	1,4	1	1	1	2	1,5	1,5	1,5	1,75	1,2	2	1,25	1,4	98
1	1,2	1,2	1	1	1	1,5	1,75	1,5	1,5	1,25	1	2	1,25	1,2	91
1	1,4	1	1	1,25	1	1,5	1,75	1,5	1,75	1,25	1,6	2,2	1,5	1,4	91
1	1,2	1,2	1	1,25	1,25	2	1,75	1,5	1,75	2	1,6	2	1,5	1,4	90
1	1,2	1,8	1	1	1,25	1,5	2	1,75	2,25	1,25	1,4	2	1,25	1,2	90
1	1,4	1,6	1	1	1	1,5	2	2	2	1,25	1,2	2,2	1,25	1,2	88
1	2,4	1,6	1,25	1,25	1,5	2,5	1,75	1,5	2,25	1,75	1,2	2	1,25	1,4	88
1	1,4	1,6	1	1,25	2,25	2,5	1,75	1,75	2,75	1,5	1,4	2	1,5	1,6	80
1	1,8	2,2	2	1,75	1,75	2,25	1,75	2,75	2	1,5	1	2,2	1,25	2	79
1	1,8	2	1,75	1,75	1,75	2,75	2	2,5	2,5	1,25	1,2	2,2	1,25	1,8	77
1	2,2	2	2,5	1	2,25	2,25	3	2,5	2,75	1,5	1,2	2,4	1,25	1,8	74
1	2	1,6	1,75	1,75	2,25	2,75	3,25	2,25	2,5	1,75	1,2	2	1,5	1,6	70
1	2,4	2,8	2	2,5	2	2,25	3,25	2,75	2,5	2	2,2	2,4	1,5	2	69
1,25	2,4	2,4	2,25	1,75	1	3,5	3,25	2	2,75	2,25	1,2	2,8	2	1,6	65
1	2,2	1,8	2	1,75	2,75	4	4	2,25	2,5	2,75	1,2	2,8	2	1,8	61
1	2,8	1,8	2	1,75	2,75	4	4	2,5	2,5	3	1,2	2,6	2	2,2	57
1	3,2	1,6	2,5	2,5	1,5	4	4	2,25	3	1,75	1,2	2,8	2,25	2	54
1,75	2,8	1,8	2,25	2,75	1,25	4	4	2,75	3,5	1,75	1,8	2,2	2,25	2,2	52
2	2,8	2	2,5	2,5	1,5	4	4	2,75	3,25	1,75	2,2	2,4	2,25	1,8	52
1,25	3,4	1,8	2,75	2,5	2,25	4	4	3	3	2	1,8	3,2	3	2,2	50
1,75	3,8	2,4	2,75	2,5	2,5	3,75	4	3,25	2,25	2,5	2,4	3,2	2,5	2,8	50
1,75	3,6	2,6	2,75	3	2,5	3,75	4	3,25	2,5	2,75	2,4	3,2	2,5	2,8	50
2	3,6	2,4	2,75	2,75	2,75	3,75	4	3,25	3	3,25	2,6	3,2	2,5	2,8	49
1,25	3	3,6	2	2	1,75	4	3,75	3,25	2,75	2	2	2,2	3	2	49
1,25	3,8	2,4	1,5	2	1,75	4	3,5	3,5	2,75	3	2	2,4	3,25	2,6	48
1,75	3	2,4	3,25	2,5	2	4	4	2,75	3	2,5	2,2	3	1,75	2,6	48
2,25	3,6	2,6	2,75	3,25	2,5	4	4	3	2,75	3	2,4	3,6	2	2,8	48
2,25	3,8	2,6	3,5	4	2,25	3,75	3,75	3,5	2,5	2,25	3,2	3,2	2,5	2,6	47
1,75	3,4	3,2	2,75	2,75	2,75	4	4	3,25	2,75	2,5	2	3,4	3	2,4	46
2	3,6	3	3,5	3,75	2,5	3,5	3,75	3,5	2,25	2,5	3,6	3	3	3,2	46
2,25	3,8	2,6	3,5	3,25	3,25	4	4	3,25	3,25	3,25	2	3,6	2,5	3	45
2,25	3,8	2,8	4	4	2,75	4	3,75	3,5	3	2,75	3,4	3,4	3	3	45
1,25	3,2	2,8	2,25	2,25	1,75	3,75	3,75	3,75	3	2,25	1,8	2,6	2,75	2,2	45
1	2,4	2,4	1,75	2,5	2,25	3,75	3	4	2,5	2,25	2,4	2,4	3	2,4	43
1	3,4	2,2	3	2,25	3	4	4	3,5	3	2,5	2,2	2,6	3	2,2	41
1,5	3	1,8	2,75	3,25	2	3,75	3,75	3,75	2,75	3	1,8	3,4	2,5	2,8	40
1,75	3,6	1,6	2,5	2	1	3,5	3,5	3,75	3	2	2,2	2,6	2,25	2,2	39
1	2,6	3	2,75	2,75	2,75	4	3,75	4	2,75	3,25	2,4	2,6	3	3,2	38
2	3,8	2,6	2,5	2,75	2	3,5	3	4	3	3,25	2,8	2,2	2	2	36
2	3,8	2,6	3,25	2,75	1,5	3,5	3,75	4	3	3,25	2,8	2,4	3,25	2	36
2	3,8	2,8	4	2,5	1,75	3,75	3,75	4	2,5	2,5	2,6	2,8	3,5	2,4	34
2	3,8	2,8	3,5	2,75	2,5	3,75	3,75	4	3	3,5	2,8	2,4	3,5	2,4	33
2	3,8	3	3,5	3	3,5	4	3,75	4	3	3	2,8	3	3,25	2,4	32
2	3,6	3,2	2,75	3,25	2,25	3,75	4	4	3,25	3,5	2,6	2,8	3	2,2	32
2,5	3,8	3,4	3,25	3,25	2,75	4	4	3,75	3,5	2,75	3	3,4	3,25	3	31
1,75	3,2	3,2	3,75	3,75	4	4	4	4	3	3	3	3,6	3	3,2	30
1,75	3,2	3,4	3,75	3,75	4	4	4	4	3	3,25	3	3,6	3	3,4	30
2,75	3,6	3	3,25	4	2,75	4	4	4	3,25	2,75	3	4	3,25	2,8	30
2,5	3,8	3,4	4	2,75	2,5	4	4	4	3	3	2,8	2,8	3,75	3,2	29
2,75	3,6	3,4	4	3,5	3,5	4	4	4	3,75	3	3	3,6	3,25	2,6	29
2,75	3,6	3,2	3,75	3,5	2,25	4	4	4	3,75	3,25	2,6	3,2	3,5	3,2	28
2,5	3,6	3,4	3,75	3	3	4	4	4	3,5	3	2,2	3,6	4	3,2	27
1,75	3,4	3	4	4	4	4	3,75	4	3,75	3,25	3	3,6	3,75	3,2	26
3	3,6	3,4	3,5	2,5	3,5	4	4	4	3,5	3	3,2	3,8	4	3	26
3	3,8	3,2	3,75	4	3	4	4	4	3,25	2,75	3,4	3,8	3,75	2,8	26
2,5	3,4	3,4	4	4	4	4	3,75	4	4	3,5	3,2	3,6	3,75	2,8	25
3,25	3,8	3,4	4	3,25	3,25	4	4	4	3,25	3	3,4	3,8	3,5	3,2	25

Figure H.2: Profile of optimisation levels for all tasks (columns) and Pareto optimal solutions (rows) of optimisation 1

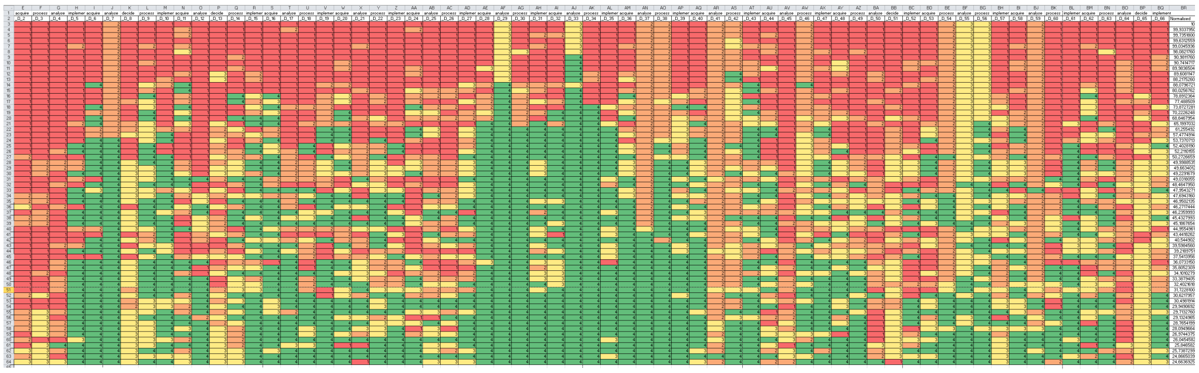


Figure H.3: Profile of optimisation levels for all activities (columns) and Pareto optimal solutions (rows) of optimisation 1

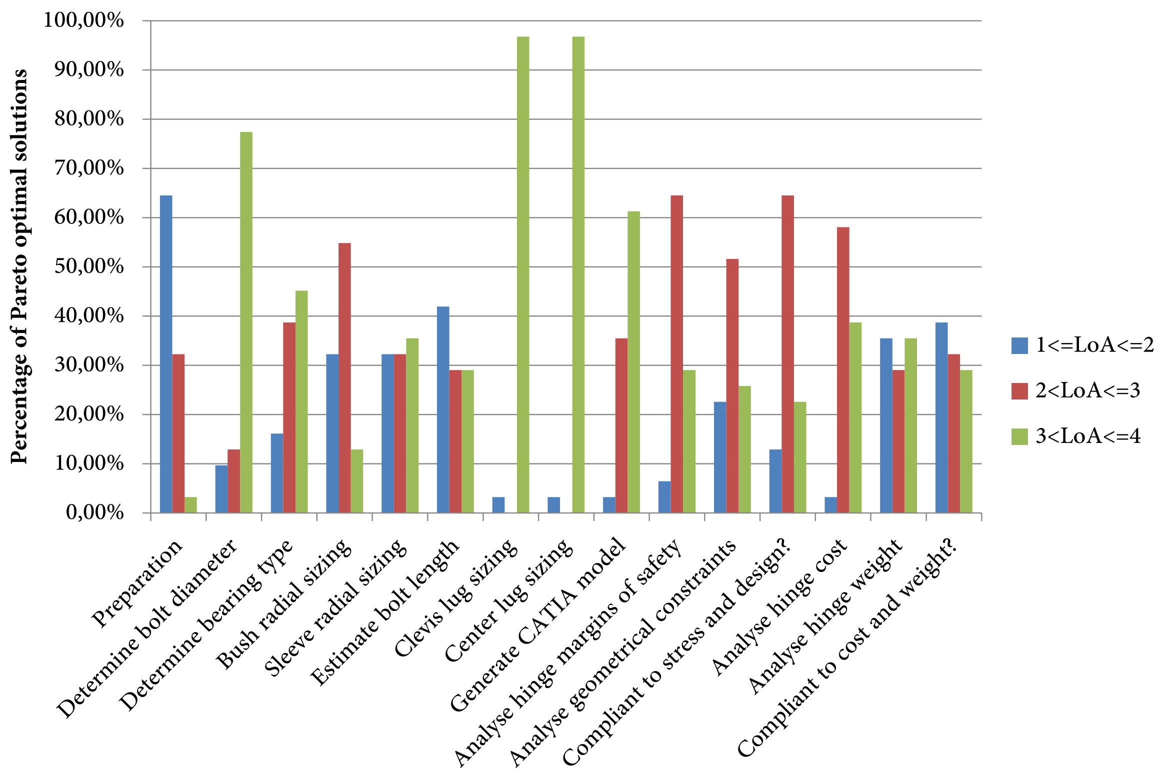


Figure H.4: Summary of the different levels of automation per task for optimisation 2

Preparation	Determine bolt diameter	Determine bearing type	Bush radial sizing	Sleeve radial sizing	Estimate bolt length	Clevis lug sizing	Center lug sizing	Generate CATIA model	Analyse hinge margins of safety	Analyse geometrical constraints	Compliant to stress and design?	Analyse hinge cost	Analyse hinge weight	Compliant to cost and weight?	Normalised lead time
1	1,2	1	1	1	1	1,5	1,5	1,5	1,5	1,25	1	2	1,25	1	100,0
1	1,6	1,8	1,75	2	1,5	3,75	4	2,25	2,75	1,5	1,6	2,2	2	1,8	61,2
1	2,6	2,4	1,5	2,25	1	4	4	2,5	2,5	1,75	2	2,2	2	1,4	55,4
1,25	2,2	2	1,5	2	1,75	4	4	2,5	2	2	2	2,8	2	1,6	55,1
1,75	2	2,2	1	1,5	2	3,75	4	3,25	2,25	1,75	2,4	2,6	2	1,8	55,0
1,75	2,4	1,4	1,75	1,75	1,5	4	4	2,75	2,25	2	2,8	3,2	2	1,6	53,3
1,75	2,6	1,6	2	2,25	2	4	4	3	2,25	2	2,4	3,2	2	2,4	52,4
1,75	3,4	2,6	2	2	2,5	4	4	3	2,25	2,5	2,6	3	1,75	2	49,5
1,75	3,4	3,2	1,75	2	2,5	3,75	3,75	2,75	3,25	2,75	3	2,4	2	2,4	48,8
1,75	3,8	2,8	2,5	1,5	3,25	3,75	3,75	2,5	3	2,5	3	2,8	2,25	2	48,1
2	3,6	3,4	2,75	2,25	3	3,75	4	3	2,75	2,5	2,6	3	2,5	1,8	46,9
2	3,4	2,8	2,25	3	2,5	4	3,75	3	3,25	2,75	3	2,8	2,5	3,2	46,8
1,75	3,4	2,6	2,75	2,75	3,75	3,5	4	3,5	3,25	2,75	3,2	3	4	2,4	44,8
2	3,4	3,2	3	2,5	3,5	4	4	2,75	3	2,75	2,8	3,6	3,75	2,6	44,1
2,5	3,4	3,6	3,25	3,25	2	4	4	3,5	2,75	3,25	2,8	3	2	3,4	43,1
1,75	3,4	3,4	2	2,25	2	3,75	4	3,5	2,5	2,75	3	2,2	2,5	1,8	37,2
2	3,6	2,6	2,25	2,5	2	3,75	3,75	3,75	3,25	2,5	2,8	2,8	2,5	2	36,6
1,75	3,4	3,6	2,25	2	1,75	4	4	3,5	2,75	2,75	2,4	3	2,75	1,8	35,6
1,75	3,4	3,8	2,5	2,75	2,5	3,75	4	3,5	2,75	3	3	2,2	2	2,2	34,8
2	3,6	2,8	2,75	2,25	3	4	4	3,25	3	3	3	3,6	3,25	2,6	32,8
1,75	3,8	3	2,25	2	3,25	4	4	3,25	2,5	3	3	2,8	3,25	2,2	32,8
2,25	3,8	2,8	2,75	3,25	2,5	4	4	4	2,5	2,75	3,2	3	2,5	3,2	32,8
2,25	3,6	3	2,25	4	2	4	3,75	3,75	2,75	3,25	3	3,4	3,25	2,8	32,3
2,25	3,8	2,8	2,75	3,75	2,5	4	3,75	4	3,25	3,5	3,6	3,2	3	3,6	30,6
2,25	3,8	3,4	2,75	3,5	3,25	4	4	4	2,75	3	3,6	3	3,5	2,8	29,1
2,5	3,8	3,4	2,5	3,5	2,25	4	4	4	3	3,25	3	3,2	2,75	3,4	28,4
2,5	3,8	3,4	2,75	3,75	2	4	4	4	2,75	3,25	2,8	3,4	3,75	3,2	27,7
2,75	3,8	3,6	3	4	3,25	4	4	3,25	3,75	3,5	3,2	3,4	4	3	26,1
3	3,6	3,8	3,5	4	4	4	4	4	3,25	3	2,8	3,4	3,75	3,6	24,8
2,75	3,8	3,8	3,75	4	3,75	4	4	4	3,75	3,75	3,4	3,8	3,75	3,8	22,2
3,5	3,8	3,8	3,25	3,5	3,25	4	4	4	3,5	3,75	3,6	3,8	3,5	3,2	22,0

Figure H.5: Profile of optimisation levels for all tasks (columns) and Pareto optimal solutions (rows) of optimisation 2

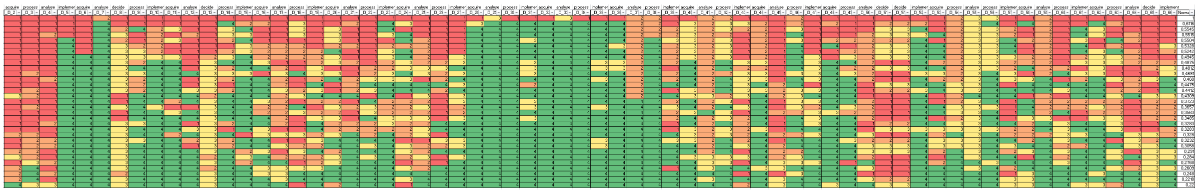


Figure H.6: Profile of optimisation levels for all activities (columns) and Pareto optimal solutions (rows) of optimisation 2

I
Optimisation settings: Hinge connection

The results of the case study are obtained using the Optimus platform. The optimisation settings for the two optimisations presented in Section 8.3 are displayed in Table I.1 and I.2 for the MOO of lead time vs investment cost and lead time vs number of projects until BEP respectively.

Table I.1: Optimisation settings for the MOO trading off lead time and investment cost

Optimisation parameter	**Description**
Target front size	80
Population size	80
Start population	No method selected
First population size	40.
Random seed	60
Weighting factor	0,7
Inverse crossover probability	0,85
Maximum number of iterations	70

Table I.2: Optimisation settings for the MOO trading off lead time and number of projects until BEP

Optimisation parameter	**Description**
Target front size	80
Population size	80
Start population	No method selected
First population size	40
Random seed	55
Weighting factor	0,7
Inverse crossover probability	0,85
Maximum number of iterations	70

Bibliography

[1] W. Verhagen, P. Bermell-Garcia, R. E. van Dijk, and R. Curran, *A critical review of Knowledge-Based Engineering: An identification of research challenges,* Advanced Engineering Informatics **26**, 5 (2012).

[2] A. Tripathy and S. D. Eppinger, *Organizing Global Product Development for Complex Engineered Systems,* IEEE Transactions on Engineering Management **58**, 510 (2011).

[3] A. Yassine and D. Braha, *Complex Concurrent Engineering and the Design Structure Matrix Method,* Concurrent Engineering **11**, 165 (2003).

[4] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, *A fast and elitist multiobjective genetic algorithm: Nsga-ii,* IEEE Transactions on Evolutionary Computation **6**, 182 (2002).

[5] B. de Vrught, *A value modelling method to assess the operational impact of automation within industry: IMPROVE*, Master's thesis, Delft University of Technology (2014).

[6] S. Brown and K. Eisenhardt, *Product development: past research, present findings, and future directions,* Academy of management review **20**, 343 (1995).

[7] D. Reinertsen, *The Principles of Product Development Flow: Second Generation Lean Product Development* (Celeritas, 2009).

[8] G. La Rocca, *Knowledge Based Engineering techniques to support aircraft design and optimization*, Ph.D. thesis, Delft University of Technology (2011).

[9] C. Terwiesch, C. H. Loch, and A. D. Meyer, *Exchanging preliminary information in concurrent engineering: Alternative coordination strategies,* Organization Science **13**, 402 (2002).

[10] K. B. Clark and T. Fujimoto, *Product development performance: Strategy, organization, and management in the world auto industry* (Harvard Business Press, 1991).

[11] E. J. Schut, *Conceptual design automation*, Ph.D. thesis, Delft University of Technology (2010).

[12] T. B. Sheridan and R. Parasuraman, *Human-automation interaction,* Reviews of human factors and ergonomics **1**, 89 (2005).

[13] W. J. Verhagen, B. de Vrught, J. Schut, and R. Curran, *A method for identification of automation potential through modelling of engineering processes and quantification of information waste,* Advanced Engineering Informatics (2015).

[14] S. Cho and S. Eppinger, *A Simulation-Based Process Model for Managing Complex Design Projects,* IEEE Transactions on Engineering Management **52**, 316 (2005).

[15] E. J. Schut, S. Kosman, and R. Curran, *A value scan methodology to improve industrial operations,* in *Proceedings of the 19th ISPE international conference on concurrent engineering* (2013) pp. 411–423.

[16] *Idealism project page,* ("2015"), accessed: 2015-01-07.

[17] V. Krishnan and K. T. Ulrich, *Product Development Decisions: A Review of the Literature,* Management Science **47**, 1 (2001).

[18] N. Joglekar and A. Yassine, *Performance of coupled product development activities with a deadline,* Management Science **47**, 1605 (2001).

[19] T. Browning and S. Eppinger, *Modeling impacts of process architecture on cost and schedule risk in product development,* Engineering Management **49**, 428 (2002).

[20] D. Reinertsen, *Lean thinking isn't so simple,* Electronic design **47**, 48H (1999).

[21] M. Li, Y. Yang, J. Bai, and X. Qin, *A framework for integrated optimization of Product Development Process,* in *proceedings of the international Conference on Research and Practical Issues of Enterprise Information Systems*, Vol. 255 (2008) pp. 1325–1334.

[22] V. Krishnan, S. D. Eppinger, and D. E. Whitney, *Model-Based Framework to Overlap Product Development Activities,* Management Science **43**, 437 (2014).

[23] M. van Tooren, G. La Rocca, and T. Chiciodean, *Advanced Design Methodologies* (University Press, 2009).

[24] H. Abdelsalam and H. Bao, *A simulation-based optimization framework for product development cycle time reduction,* IEEE Transactions on Engineering Management **53**, 69 (2006).

[25] Q. Yang, T. Yao, T. Lu, and B. Zhang, *An Overlapping-Based Design Structure Matrix for Measuring Interaction Strength and Clustering Analysis in Product Development Project,* IEEE Transactions on Engineering Management **61**, 159 (2014).

[26] R. Brandao and M. Wynn, *Improving the New Product Development Process through ICT Systems in the Aerospace Industry: A Report on Case Study Research,* in *proceedings of the 2009 International Conference on Information, Process, and Knowledge Management* (2009) pp. 147–152.

[27] Z. Wu, L. Li, and H. Zhao, *Simulation and analysis of schedule and cost of product development,* in *proceedings of the 2010 International Conference on Mechanical Automation and Control Engineering, Wuhan, China* (2010) pp. 2–7.

[28] J. A. Reed, G. J. Follen, and A. A. Afjeh, *Improving the aircraft design process using Web-based modeling and simulation,* ACM Transactions on Modeling and Computer Simulation **10**, 58 (2000).

[29] T. R. Browning, *Use of Dependency Structure Matrices for Product Development Cycle Time Reduction,* in *Fifth ISPE International Conference on Concurrent Engineering: Research and Applications, Tokyo, Japan* (1998) pp. 1–8.

[30] C. T. Fitz-Gibon, *Performance Indicators* (WBC Print Ltd, Bristol, 1990).

[31] P. Adler and A. Mandelbaum, *Getting the most out of your product development process,* Harvard Business Review **March-April**, 1 (1996).

[32] T. R. Browning and R. V. Ramasesh, *A Survey of Activity Network-Based Process Models for Managing Product Development Projects,* Production and Operations Management **16**, 217 (2007).

[33] S. D. Eppinger, D. E. Whitney, R. P. Smith, and D. A. Gebala, *A model-based method for organizing tasks in product development,* Research in Engineering Design **6**, 1 (1994).

[34] M. R. Millson, S. P. Raj, and D. Wilemon, *A Survey of Major Approaches for Accelerating New Product Development,* Journal of Product Innovation Management **9**, 53 (1992).

[35] W. van der Aalst and A. ter Hofstede, *YAWL: Yet Another Workflow Language,* Information Systems **30**, 245 (2005).

[36] R. Smith and S. Eppinger, *A predictive model of sequential iteration in engineering design,* Management Science **43**, 1104 (1997).

[37] S. Ha and H.-W. Suh, *A timed colored Petri nets modeling for dynamic workflow in product development process,* Computers in Industry **59**, 193 (2008).

[38] M. Hammer, *Seven insights about processes,* in *Proceedings of the Conference on Strategic Power Process Ensuring Survival Creating Competitive Advantage, Boston, MA, US* (2001).

[39] T. Browning, *Applying the design structure matrix to system decomposition and integration problems: a review and new directions,* IEEE Transactions on Engineering Management **48**, 292 (2001).

[40] C. van der Velden, C. Bil, and X. Xu, *Adaptable methodology for automation application development,* Advanced Engineering Informatics **26**, 231 (2012).

[41] K. Clark, W. Chew, and T. Fujimoto, *Product development in the world auto industry,* Brookings Papers on Economic Activity **1987**, 729 (1987).

[42] J. J. Hart and J. Valasek, *Methodology for prototyping increased levels of automation for spacecraft rendezvous functions* (Texas A&M University, 2010).

[43] Å. Fasth, J. Stahre, and K. Dencker, *Measuring and analysing levels of automation in an assembly system,* in *Manufacturing Systems and Technologies for the New Frontier* (Springer, 2008) pp. 169–172.

[44] I. Balogh, K. Ohlsson, G.-Å. Hansson, T. Engström, and S. Skerfving, *Increasing the degree of automation in a production system: consequences for the physical workload,* International Journal of Industrial Ergonomics **36**, 353 (2006).

[45] R. Parasuraman, T. Sheridan, and C. Wickens, *A model for types and levels of human interaction with automation,* IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans **30**, 286 (2000).

[46] C. A. Miller and R. Parasuraman, *Beyond levels of automation: An architecture for more flexible human-automation collaboration,* in *Proceedings of the Human Factors and Ergonomics Society Annual Meeting,* Vol. 47 (SAGE Publications, 2003) pp. 182–186.

[47] C. Wickens, *Stages and Levels of Automation: An Integrated Meta-analysis,* in *The human factors and ergonomics society 54th annual meeting,* Vol. 4 (2010) pp. 389–393.

[48] R. Parasuraman, *Designing automation for human use: empirical studies and quantitative models,* Ergonomics **43**, 931 (2000).

[49] C. L. Emberey, N. Milton, J. Berends, M. Van Tooren, S. Van der Elst, and B. Vermeulen, *Application of knowledge engineering methodologies to support engineering design application development in aerospace,* in *proceedings of the 7th AIAA Aviation Technology, Integration and Operations Conference (ATIO)* (2007).

[50] N. R. Milton, *Knowledge acquisition in practice: a step-by-step guide* (Springer Science & Business Media, 2007).

[51] N. Srinivas and K. Deb, *Multiobjective optimization using nondominated sorting in genetic algorithms,* Evolutionary computation **2**, 221 (1994).

[52] K. Müller and T. Vignaux, *SimPy: Simulating Systems in Python,* (2003).

[53] *Optimus Theoretical Background,* Noesis Solutions, 10th ed. (2014).

[54] R. P. Smith and J. A. Morrow, *Product development process modeling,* Design Studies **20**, 237 (1999).