CHALLENGES IN EUROPEAN AEROSPACE
5TH CEAS AIR & SPACE CONFERENCE
CEAS

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.

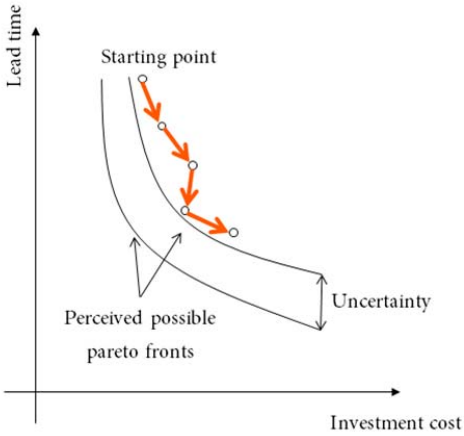

Figure 1: Current trade-off between cost and lead time in a PDP

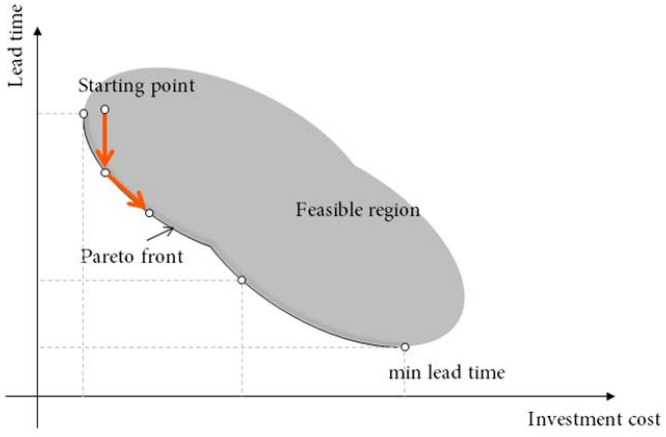

Figure 2: Improved trade-off between cost and lead time in a PDP

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.

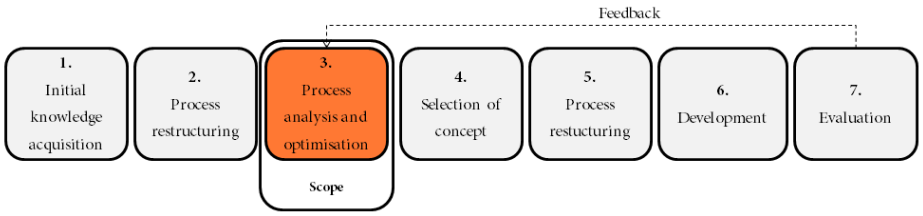

Figure 3: Steps involved in the proposed methodology

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 $\boldsymbol{z}$.

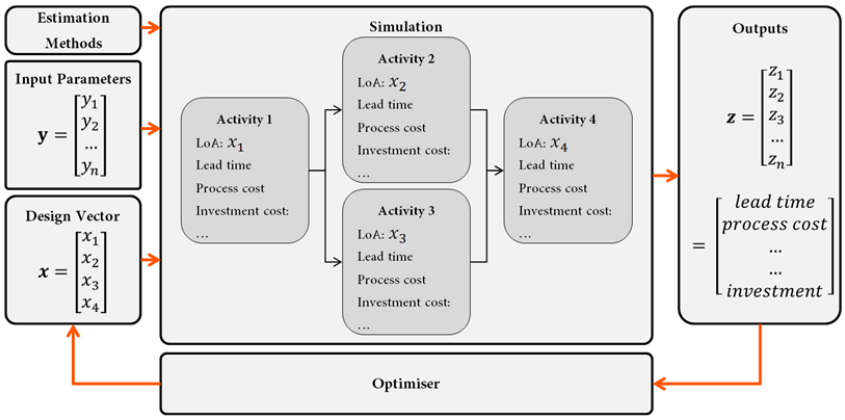

Figure 4: Overview of the proposed PDP simulation and optimization framework

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.

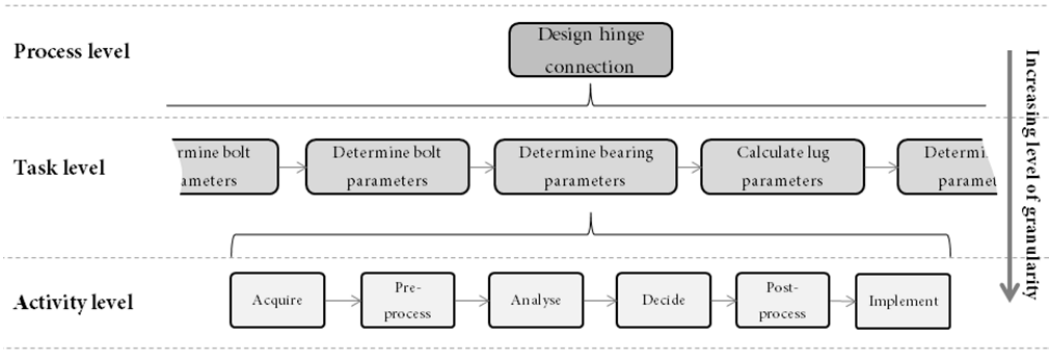

Figure 5: Decomposition from process level to activity level

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.

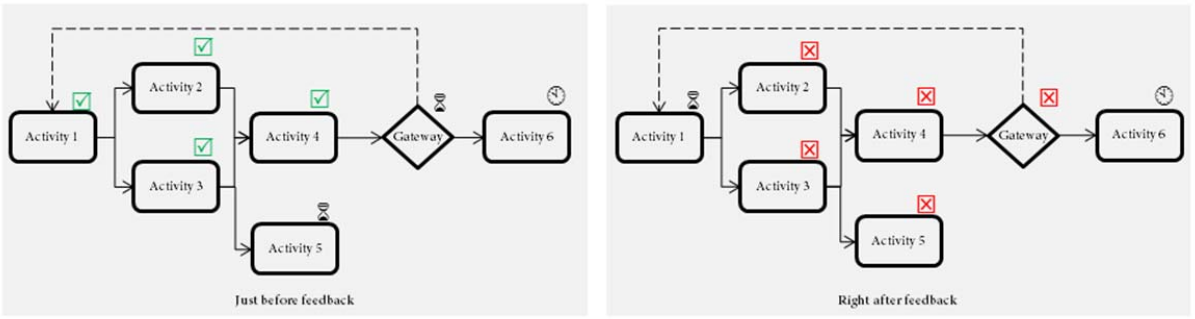

Figure 6: Illustration of rework policy

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).

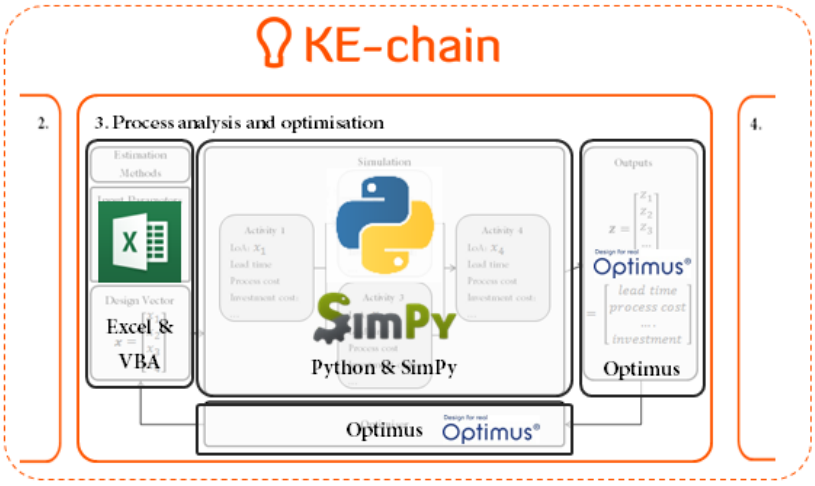

Figure 7: Integrated framework overview as part of the full methodology illustrated in Figure 3

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.

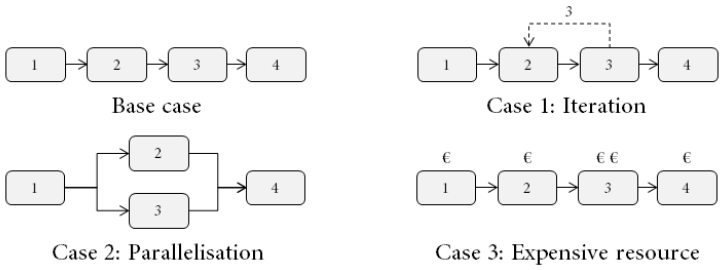

Figure 8: Flowcharts of multiple test cases

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.

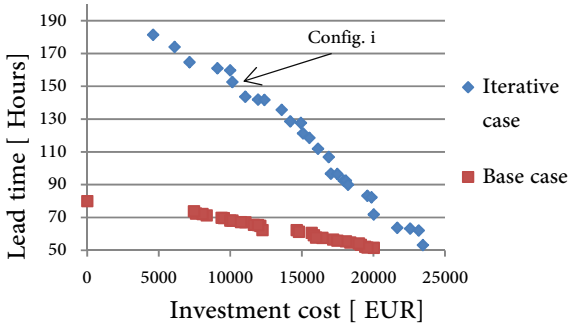

Figure 9: Pareto front for the lead time and investment cost for the base case and iterative case

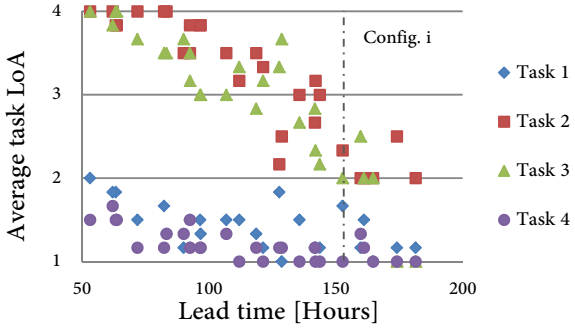

Figure 10: Average level of automation per task

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.

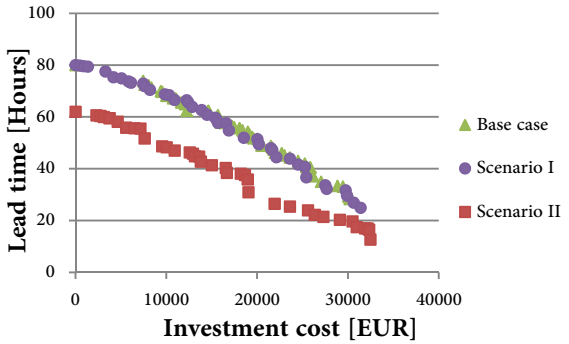

Figure 11: Pareto front for lead time and investment cost for parallelisation cases and base case

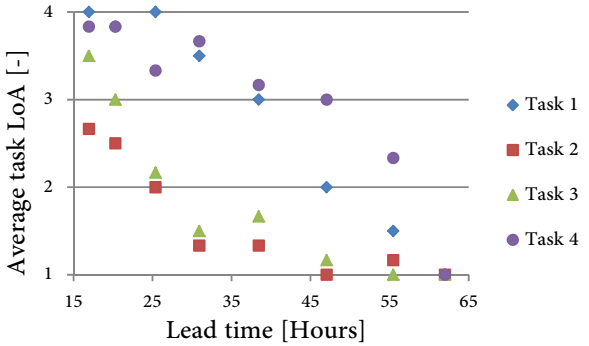

Figure 12: Selection of architectures on the Pareto front for Scenario II with average levels of automation

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.

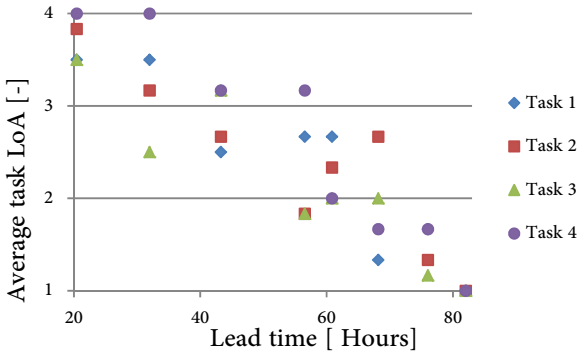

Figure 13: Average level of automation per task for base case for configuration on the Pareto front

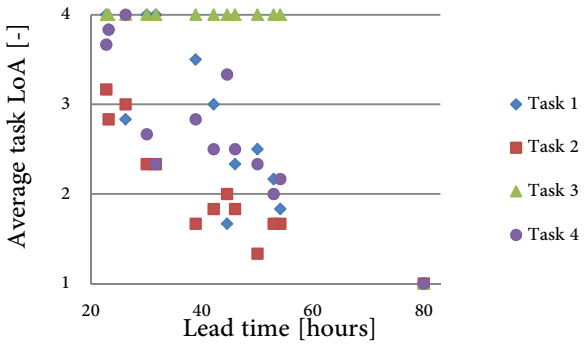

Figure 14: Average level of automation per task for test case 3 for configurations on the Pareto front

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

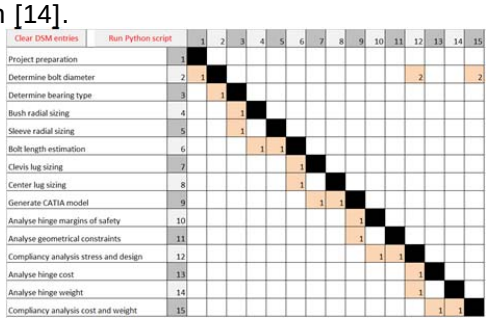

Figure 16: Activity-based Design Structure Matrix of the case study

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.

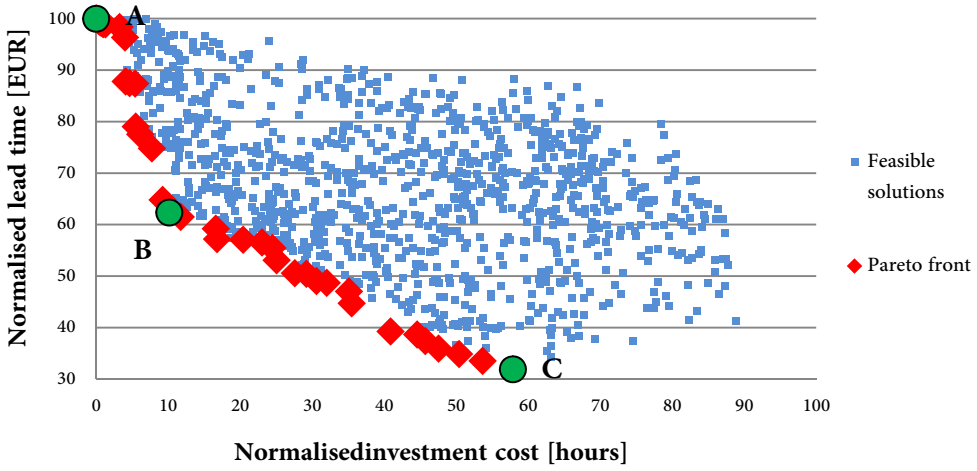

Figure 17: Pareto front for the Multi-Objective Optimisation for lead time and total investment cost

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.