TECHNISCHE UNIVERSITÄT MÜNCHEN

Lehrstuhl für Raumfahrttechnik

Time-Cost Tradeoffs in Product Development Processes

Christoph Meier

Vollständiger Abdruck der von der Fakultät für Maschinenwesen der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs (Dr.-Ing.)

genehmigten Dissertation.

Die Dissertation wurde am 09.02.2011 bei der Technischen Universität München eingereicht und durch die Fakultät für Maschinenwesen am 20.06.2011 angenommen.

Acknowledgements

At this point, I would like to thank many people who supported me during my entire Ph.D. studies and who eventually contributed to the production of the underlying thesis. First of all, my deepest gratitude is to my advisor, Prof. Ulrich Walter, who has been very generous to me over the last couple of years. In particular, I am very thankful for the freedoms he gave me. My co-advisors Prof. Browning and Prof. Yassine have been closely collaborating with me for many years and actually convinced me to pursue a Ph. D. degree back in 2006. It is not only their subject-specific input and feedback I would like to honor but also their personal advices and kindness. Additionally, I would like to thank Prof. Lindemann for taking the chair of the committee. Beside my three advisors, Prof. Olivier de Weck at MIT supervised me for an exciting year in Cambridge and I still love to look back on the work with him and his research group. I also appreciate the helpful and always interesting discussions on any topic with Prof. Igenbergs. Furthermore, many thanks are due to my colleagues and to the entire staff at the institute of astronautics who made the office a lovely place to work. Specifically, I like to thank Mrs. Lochner for her administrative aid. Last but not least, and most importantly, I am grateful to my family, friends and girl-friend who steadily encouraged me to finalize my dissertation. This work was sponsored by the *Bavarian Science* foundation.

Abstract

While rapid and low-cost product development (PD) processes have been identified as a primary success factor for companies in highly competitive markets, the simultaneous reduction of development cost and lead-time is not possible. In reality, a time-cost tradeoff situation arises as organizations seek the fastest PD process subject to a predefined budget, or vice-versa, identifying the lowest cost PD process for a given project duration. Identifying the set of all best time-cost tradeoff solutions constitutes a fundamental optimization problem in the engineering- and operations management literature. It is of great practical relevance since PD processes may involve a substantial amount of costly resources employed over an extended period of time. Despite its thorough study in past decades, previous research was based on limiting assumptions, which are challenged in this thesis. Thus, the underlying work intends to overcome previous research deficits in order to derive new managerial insights which contribute to our improved understanding of PD processes and ultimately better products, too.

For this purpose, we first introduce a new model for PD processes simultaneously accounting for cyclic process architectures, overlapping, crashing and work policy. These four process parameters can be considered as the most fundamental as they considerably affect process time/cost. Past research on the time-cost tradeoff problem in PD merely considered one or two of these parameters jointly, but not all four. Next, we conduct a sensitivity analysis of this complex model using simulation in order to examine the effects of the aforementioned parameters on process time/cost. Subsequently, we propose an optimization strategy capable of identifying the set of all best time-cost tradeoffs solutions for a PD process. Finally, a PD process in the aerospace industry as well as a process in the automotive industry served as case study for our new model, simulation and optimization approach thereby verifying insights derived from artificially constructed processes throughout the thesis.

Among other insights and findings, this work reveals that time-cost tradeoffs in PD may not only emerge from modifications of process architecture, crashing- or overlapping strategy – which is the current state of knowledge – but also from work policy rules. Furthermore, our investigation indicates that feedbacks economically favor the application of crashing and overlapping. To some extent, we hence claim that the temporal benefits of process time cutting strategies in PD outweigh their cost related drawbacks with increasing process dynamics. Also, we discovered that increasing parallel information flows within a PD process economically foster the application of crashing and overlapping. Last but not least, we could derive scale-up behaviors for PD process time and cost as a function of various parameters.

CONTENTS

0. Nomenclature

0.1 Abbreviations

AoA	**A**ctivity-**o**n-**A**rc network
AoN	**A**ctivity-**o**n-**N**ode network
CE	**C**oncurrent **E**ngineering
CPM	**C**ritical **P**ath **M**ethod
DSM	**D**esign **S**tructure **M**atrix
GA	**G**enetic **A**lgorithm
GERT	**G**raphical **E**valuation and **R**eview **T**echnique
LHS	**L**atin **H**ypercube **S**ampling
MOEA	**M**ulti-**O**bjective **E**volutionary **A**lgorithm
MOGA	**M**ulti-**O**bjective **G**enetic **A**lgorithm
MOO	**M**ulti-**O**bjective **O**ptimization
MOOP	**M**ulti-**O**bjective **O**ptimization **P**roblem
PD	**P**roduct **D**evelopment
PDF	**P**robability **D**ensity **F**unction
PERT	**P**rogram **E**valuation and **R**eview **T**echnique
SOO	**S**ingle-**O**bjective **O**ptimization
SOOP	**S**ingle-**O**bjective **O**ptimization **P**roblem
UCAV	**U**nmanned **C**ombat **A**erial **V**ehicle
WBS	**W**ork **B**reakdown **S**tructure

0.2 Symbols

M_1	Design Structure Matrix with rework probabilities
M_2	Design Structure Matrix with rework impacts
M_3	Design Structure Matrix with minimal overlapping available intensities without

	rework penalty
M_4	Design Structure Matrix with maximal overlapping needed intensities without rework penalty
M_5	Design Structure Matrix with minimal overlapping available intensities
M_6	Design Structure Matrix with maximal overlapping needed intensities
$\boldsymbol{F^*}$	Pareto front
$\boldsymbol{F^*}_{\varepsilon}$	Pareto front containing only ε-dominated vectors
$\boldsymbol{X}^*$	Decision space
$\boldsymbol{Y}^*$	Objective space
$\boldsymbol{A}$	Decision vector
$\boldsymbol{B}$	Decision vector
$\boldsymbol{C}$	Vector with sampled cost values for every activity
$\boldsymbol{D}$	Vector with sampled duration values for every activity
$\boldsymbol{H}$	Vector with active activities during process simulation
$\boldsymbol{I}$	Vector with inactive activities during process simulation
$\boldsymbol{L}$	Vector with learning curve values for cost and duration for every activity
$\boldsymbol{P}$	Vector containing a permutation encoded as random keys
$\boldsymbol{P_i}$	Vector containing all chronologically precedent activities of activity *i*
$\boldsymbol{R}$	Vector with maximal crashing intensities for every activity
$\boldsymbol{S}$	Vector that accounts for the activity sequence in the process
$\boldsymbol{X}$	Decision vector
$\boldsymbol{Y}$	Objective vector
c_i	Sampled cost of activity *i*
$\hat{c}_i$	Maximal cost of activity *i*
$\bar{c}_i$	Most likely cost of activity *i*
$\breve{c}_i$	Minimal cost of activity *i*
c_{tot}	Total process cost considering crashing cost and overlapping cost
$c_{tot/c}$	Total process cost without consideration of crashing cost
$c_{tot/o}$	Total process cost without consideration of overlapping cost
$c_i\left(k_i\right)$	Actual cost of activity *i* in the k_i-th iteration of activity *i*
$\tilde{c}_i\left(k_i\right)$	Actual cost of activity *i* in the k_i-th iteration of activity *i* considering rework impact and learning effects
$\Delta c_{C_i\left(k_i\right)}$	Change in cost of activity *i* due to crashing in the k_i-th iteration of activity *i*

$\Delta c_{O_i(k_i)}$	Change in cost of activity *i* due to an overlapping with all chronological predecessors in the k_i-th iteration of activity *i*
$\Delta c_{R_i(k_i)}$	Change in cost of activity *i* in the k_i-th iteration due to partial rework
$\Delta c_{c_{tot}/c}$	Change in cost in percent of the total process cost due to crashing
$h_{ij}\left(t_{O_{ij}(k_i,k_j)}\right)$	Overlapping function: duration of rework due to an overlapping between activity *i* and *j* in the k_i-th iteration of activity *i* and the k_j-th iteration of activity *j*
$\hat{h}_{ij}\left(t_{O_{ij}(k_i,k_j)}\right)$	Maximal duration of rework due to an overlapping between activity *i* and j in the k_i-th iteration of activity *i* and the k_j-th iteration of activity *j*
k_i	Iteration number for activity *i*
$\hat{k}_i$	Maximal number of iterations allowed for activity *i* provoked by any other activity
l_i	Learning curve value of activity *i* for cost and duration
n_A	Number of activities in the process
n_B	Number of bits used to encode crashing and overlapping intensities
n_R	Number of reworked activities in the process
n_{fb}	Number of feedback relationships in a process
n_{ff}	Number of feed-forward relationships in a process
$n_{r,i}$	Maximal number of partial reworks for an activity *i*
$o_{D_{ij}(k_i,k_j)}$	Percentage of activity *i*'s duration when it delivers the output in the k_i-th iteration for activity *j* in the k_j-th iteration
$o_{R_{ij}(k_i,k_j)}$	Percentage of activity *j*'s duration when it receives the output in the k_i-th iteration from activity *i* in the k_i-th iteration
p_c	Crossover probability
p_m	Mutation probability
$\hat{r}_i$	Maximal crashing intensity for activity *i*
$r_i(k_i)$	Actual crashing intensity for activity *i* in the k_i-th iteration of activity *i*
t_{tot}	Total process duration
$\hat{t}_{tot}$	Maximal process duration (out of a set of process durations)
$\hat{t}_{tot}$	Minimal process duration (out of a set of process durations)
$t_{tot/c}$	Total process duration without consideration of crashing
$t_{tot/o}$	Total process duration without consideration of overlapping
t_i	Sampled duration of activity *i*
$\hat{t}_i$	Maximal duration of activity *i*

$\bar{t}_i$	Most likely duration of activity *i*
$\breve{t}_i$	Minimal duration of activity *i*
$t_i(k_i)$	Actual duration of activity *i* in the k_i-th iteration of activity *i*
$\tilde{t}_i(k_i)$	Actual duration of activity *i* in the k_i-th iteration of activity *i* considering rework impact and learning effects
$t_{O_{ij}(k_i,k_j)}$	Overall duration of overlapping between activity *i* and *j* in the k_i-th iteration of activity *i* and the k_j-th iteration of activity *j*
$\tilde{t}_{O_{ij}(k_i,k_j)}$	Overall duration of overlapping between activity *i* and *j* considering the dates $T_{R_{ij}(k_i,k_j)}$ and $T_{D_{ij}(k_i,k_j)}$
$\Delta t_{C_i(k_i)}$	Change in duration of activity *i* due to crashing in the k_i-th iteration of activity *i*
$\Delta t_{O_i(k_i)}$	Change in duration of activity *i* due to rework caused by an overlapping between activity *i* and all its chronological predecessors in the k_i-th iteration of activity *i*
$\Delta t_{R_i(k_i)}$	Change in duration of activity *i* in the k_i-th iteration due to partial rework
$\Delta t_{t_{tot}/c}$	Change in duration in percent of the total process duration due to crashing
$R_i\left(r_i\left(k_i\right)\right)$	Crashing function generating a dimensionless value in dependence of crashing intensity in the k_i-th iteration of activity *i*
T_E	Point in time for the next event of the discrete event simulation
$\breve{T}_I$	Minimal finish time of all inactive activities
$\breve{T}_D$	Minimal point in time when an active activity is allowed to deliver output to any other activity
$T_{s_i(k_i)}$	Start time of activity *i* in the k_i-th iteration of activity *i*
$T_{f_i(k_i)}$	Finish time of activity *i* in the k_i-th iteration of activity *i*
$T_{A_{ij}(k_i,k_j)}$	Point in time in the k_i-th iteration of activity *i* when the output of activity *i* is fully available for activity *j* in the k_j-th iteration
$T_{D_{ij}(k_i,k_j)}$	Point in time in the k_i-th iteration of activity *i* when the output of activity *i* is effectively delivered for activity *j* in the k_j-th iteration
$T_{N_{ij}(k_i,k_j)}$	Point in time in the k_j-th iteration of activity *j* when activity *j* needs the input of activity *i* in the k_i-th iteration
$T_{O_{ij}(k_i,k_j)}$	Point in time in the k_i-th iteration of activity *i* and the k_j-th iteration of activity *j* to overlap activity *i* and *j*
$T_{R_{ij}(k_i,k_j)}$	Point in time in the k_j-th iteration of activity *j* when activity *j* effectively receives

	the input of activity *i*
α	Factor for the crashing function with $\alpha \in [0,\infty)$
α_{ij}	Factor for the overlapping function with $\alpha_{ij} \in [0,\infty)$
β	Coefficient for calculation of rework in case of simultaneous overlapping events with $\beta \in (-\infty,0]$
θ_i	Cumulative rework factor for activity *i*
κ_i	Brooks factor for activity *i*
λ_{ij}	Product of learning curve value and rework impact
ϕ	Crashing effect parameter which is the ratio for the time change due to crashing in percent of total process duration and the cost change due to crashing in percent of total cost
ς	Factor indicating the priority of ψ in favor of cost or duration
τ	Factor indicating the priority of ϕ in favor of cost or duration
$\upsilon_i(k_i)$	Factor for rework cost due to overlapping for activity *i* in iteration k_i
ψ	Overlapping effect parameter which is the ratio for the time change due to overlapping in percent of total process duration and the cost change due to overlapping in percent of total cost
$\varpi_i(k_i)$	Product of crashing intensity for activity *i* in iteration k_i and Brooks factor

Part I

INTRODUCTION AND BACKGROUND INFORMATION

1. Introduction

1.1 Motivation

The product development process

Organizations face tremendous challenges to gain and sustain a competitive advantage in a mass-customized, global market. In such an environment, rapid, frugal, and effective *product development* (PD) has been identified as a source of competitive advantage (Clark and Fujimoto 1991, Langerak and Hultink 2005). Consequently, PD has received extensive consideration in the literature (Krishnan and Ulrich 2001). Ulrich and Eppinger (2004) defined PD as "(…) the set of activities beginning with perception of a market opportunity and ending with the production, sale, and delivery of a product". The collection of these various activities and their *dependency relationships* (or *interfaces*) constitutes the *PD process*. Thereby, *activities* represent system elements and their corresponding information flows among each other form the system relationships. Thus, PD processes can be regarded as *systems* and therefore benefit from the application of systems thinking and the tenets of systems engineering (Browning and Eppinger 2002). It is important to note that PD processes often involve creativity and experimentation, and thus can be highly iterative – as opposed to other types of processes like production- or assembly processes – exchanging information rather than hardware components. Eppinger (2001) actually described the exchange of information as "(…) the lifeblood of product development". Consequently, the information flow within the PD process, more precisely its structure, greatly affects its cost, duration and quality since it determines the point in time when activities receive or deliver

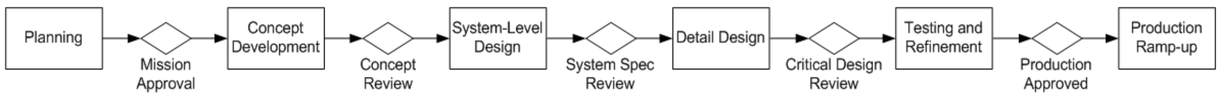


Figure 1.1: A generic product development process (adapted from Ulrich and Eppinger, 2004).

necessary information from, or to, other activities. Information flows in PD occur on a top-level between activities as well as on the bottom level within each activity (Bhuiyan 2004). Although Ulrich and Eppinger (2004) define a generic PD process (Figure 1.1) on a macro-level (i.e. process level) composed of six phases, PD processes may slightly differ even on a high-level view from firm to firm or within the same enterprise for similar projects – not to speak of a micro level (i.e. activity level) view with individual activities in place of entire development phases. According to Ulrich and Eppinger (2004), such variants of the generic PD process particularly depend on the type of product to be developed.[1]

Measures of PD success

Whereas a broad survey of PD research reveals differences in how PD is actually executed in companies (Krishnan and Ulrich 2001), the measures of PD performance are fortunately less variant. Consistent with Ulrich and Eppinger (2004) the success of PD can be mainly assessed along five dimensions: product quality, product cost, development time, development cost, and development capability. Due to our focus on the development process, we only considered development time and development cost in this thesis, taking the other three criteria as given. Both, development time and cost strongly vary with the complexity of the underlying product to be developed. For instance, the development of a screwdriver may take only one man-year of development time and about €150k development cost (Ulrich and Eppinger 2004) while the entire PD process for the Airbus A380 airplane roughly last 5 years[2]

[1]They classify products into 8 types: 1) market-pull products 2) technology-push products 3) platform products 4) process intensive products 5) customized products 6) high-risk products 7) quick-build products and finally 8) complex systems.
[2] The development history of this aircraft can be traced back to the 1980s. Though, Airbus officially announced the development in December 2000.

with thousands of engineers being involved and cost approximately €12 billion[3] when the first aircraft was completed.

Strategies to reduce PD time and cost

Many independent studies and empirical investigations have confirmed a direct relationship between the economic success of a new product and quick PD processes (Calantone and Di Benedetto 2000, Tatikonda and Montoya-Weiss 2001, Chen et al. 2005). Clearly, developing products faster than competitors allows a company to gain a larger marketing window and potentially a first-mover advantage (Lieberman and Montgomery 1988) which may be decisive for success assuming distinct market conditions.[4] Therefore, numerous PD cycle time cutting methods and tools have been proposed in the past. Most notably in this context is a systematic approach called *Concurrent Engineering* (CE). This approach reflects an engineering management philosophy composed of an entire set of principles across different management domains that jointly accelerate PD processes, decrease PD cost and improve process quality (Funk 1997, Yassine and Braha 2003). Rediscovered in the late 1970’s and early 1980’s[5], particularly in the USA, the benefits of CE can be achieved through a combination of overlapping phases, cross-functional integration and analytical tools such as design for assembly, quality function deployment or computer-aided design (Funk 1997). However, if we merely focus on the enhancements related to the PD process itself, then two distinct approaches can be identified in literature and in practice.

The first one takes a macro perspective of the development process considering the dependency relationships amongst the constituent development phases or activities. The objective is to find an optimal arrangement or sequence for these development activities based on their dependency or precedence relationships. Such an arrangement of activities and their

[3] www.manager-magazin.de (12.01.2005).
[4] Though, potential drawbacks for first-mover’s exist as well (see Lieberman and Montgomery 1988).
[5] In fact, Smith (1997) could even trace back the fundamentals of CE until the end of the 19th century.

pattern of interaction is referred to as *process structure* or *process architecture* (Browning and Eppinger 2002). An optimal arrangement provides a streamlined development process subject to a defined objective (e.g. minimal lead-time).

The second approach to accelerate development processes assumes an established process architecture and attempts to reduce lead time further through *activity crashing* and/or *overlapping*. While crashing attempts to shorten the PD process with intra-activity changes by adding resources to an activity, overlapping considers inter-activity relationships.

Time-Cost tradeoffs in PD

In fact, significant progress has been made as many firms could cut cycle time of their PD processes in half (Smith and Reinertsen 1997) by applying enhancement techniques like the ones mentioned in the previous paragraph. Nevertheless, the potential strategic advantage of expediting PD processes is typically accomplished at the expense of augmented cost. Streamlining the process architecture, adding resources to bottleneck activities and overlapping entire design phases may help to shorten lead-time but not necessarily to reduce cost. These methods may provoke *activity iteration* and *activity rework*[6] or they assign additional resources (e.g. staff) to the project which greatly affects the budget since cost in PD is mainly governed by project duration and the number of people allocated to it (Ulrich and Eppinger 2004).

Indeed, we do not want to preclude the possibility of developing PD processes which are optimal with respect to both dimensions. However, in practice, managers rather face *time-cost tradeoff* decisions when planning PD processes. In order to facilitate this decision process, they are especially interested in a specific set of processes exclusively composed of the best trade-off processes, i.e. a subset of all possible processes featuring the fastest lead-time for a certain budget. Such a set containing all best trade-off solutions is called *Pareto-*

[6] We will explain the genesis of activity iteration and rework in chapter 2.

front and its calculation for PD processes is a well-known, non-trivial, problem in project management. Nevertheless, related publications to date rely on limiting assumptions hence degrading the practical relevance of the presented results and insights. In particular, the structure of PD processes was assumed to be acyclic thereby neglecting potential activity rework. In order to provide some empirical evidence for this issue, let us briefly devote our attention to the performance of PD projects in recent years.

Cost and schedule overruns in PD

Despite all advances in project management techniques and impressive progress regarding project planning and estimation, cost and schedule (c/s) overruns for PD projects still persist across industry. Particularly c/s overruns of recent large-scale projects in the aerospace industry hit the headlines. For instance, Airbus had to spend about €2.2 billion extra money for the development of the A380 airplane which equals an overrun of 25%[7] of the initial estimates. In the USA, the Boeing Inc. officially started the development of the new "787-Dreamliner" aircraft in April 2004 optimistically predicting a development time of nearly 4.5 years. Following several announcements of delay, the first flight of the "787" occurred on December 15th, 2009, which roughly corresponds to a delay of 2 years or 50% of original schedule. Likewise, if we concentrate on the software development industry, c/s overruns seem to be a familiar phenomenon, too. The internationally recognized Standish Group releases a yearly report on software project failures with disillusioning results in 2004 (Standish Group 2004): 18% of all projects have been cancelled before they ever got completed, 53% of all projects exhibited substantial c/s overruns while merely 29% of all projects finished on budget and on schedule. Similarly, in automotive industry, c/s overruns of more than 15% occur in 60%-80% of all PD projects according to Hab and Wagner (2006). We would like to dispense further examples but instead claim that c/s overruns in PD are the

[7] www.wikipedia.org

rule rather than the exception. Obviously, the reasons for such a poor track record are manifold comprising factors like the change of system requirements, politics, inappropriate skills and low morale of employees or poorly defined objectives – just to name a few. However, poor management of the PD process and the lack of proper analysis tools is partially blamable as well (Standish Group 2004, de Weck 2006). Given the fact that PD processes exhibit some kind of repeatable structure for non-unique projects (Browning and Ramasesh 2007) and that many activities in PD are routine enough to allow their statistical characterization (Adler 1995), these overruns are even more disappointing.

Reasons for cost and schedule overruns

One of the main reasons for cost and schedule overruns is the lack of adequate PD process models which can cope with the dynamics of PD in practice. Typically, these dynamics arise as a result of iterations in the process which are often planned in order to improve quality and to allow for innovation (Eppinger 2001). Though, the immediate side effect of iterations in the process is the rework of activities increasing cost and potentially duration as well. Consequently, neglecting iterations in the PD process generally results in optimistic schedules and budget planning. In fact, rich and powerful PD process models explicitly accounting for iteration exist. But these models are usually too complex for analytical optimization methods and have therefore neither been used for a separate time/cost optimization nor for a conjoint time-cost tradeoff optimization. Rather, outdated models are widespread in literature and practical use for predicting cost and lead-time of a process as well as for process optimization. Not surprisingly, the application of these models results in unrealistic c/s predictions and suboptimal (i.e. overpriced and delayed) processes which were supposed to be "optimal"/"Pareto-optimal".

Thesis motivation

As consequence, a new, more comprehensive, investigation of time-cost tradeoffs for PD processes is necessary in order to overcome the well-known shortcomings. Therefore, this thesis explicitly addresses the impact of feedbacks on the time-cost problem in PD using realistic process models, time/cost simulation as well as state-of-the art optimization techniques. The novel, and more practical, managerial insights gained through such a study will ultimately help to design better, i.e. cheaper and faster, PD processes and consequently better products as well. Given their magnitude of cost and duration, the resulting monetary benefits may be enormous.

1.2 Scope of Work and Research Gap

Scope of work

The literature on time-cost tradeoffs of PD processes and – in particular – related research fields, e.g. PD process modeling or process optimization, is extremely vast. Therefore we limit the scope of this study as follows.

Process modeling:

With respect to process models we focus on *activity-network based models*, which view a project as a process decomposed into a network of activities. Most PD process models have used the activity network as framework (Browning and Ramasesh 2007). We neither consider *systems dynamics models* of a PD process (Ford and Sterman 1998) nor *causal models* or *parametric models* (Browning and Ramasesh 2007). System dynamics models typically take a holistic "black box" view on a process with stocks and flows of generic work to be done (Browning and Ramasesh 2007). This perspective makes it difficult to explore the effects of particular process parameters (e.g. process architecture, crashing) on time and cost which is an objective of this thesis. Also, causal models or parametric models do not appear to be appropriate for this study as they use techniques such as regression analysis (Browning and Ramasesh 2007) to derive general relationships between model parameters.

Process resources:

Furthermore, we ignored the explicit scheduling of resources for the actual analysis of time-cost tradeoffs. In fact, many resources must be managed during PD (e.g. staff or machinery) thereby posing constraints on the start & finish of an activity consequently also affecting the outcome of the time-cost Pareto-front of any process. Hence, the non-consideration of resource constraints is a limitation of our model but it is a reasonable assumption for this thesis as financial capital and duration of an activity happen to be the most fungible resources. Accounting for additional resources only delays process lead-time and/or augments cost.

Process measures:

We do not focus on the product itself therefore not only disregarding product quality but also measurements for the impact of process quality on cost and time. Analogical to the dispensation with resources, quality of a process is expected to affect its cost and time thus potentially extending time-cost tradeoffs by a further dimension resulting in time-cost-quality tradeoffs. Nevertheless, we waived an explicit investigation due to the huge research gaps in the field of time-cost-quality tradeoffs and the already ambitious objectives to be covered in this thesis. Implicitly, however, we do consider process quality because of the consideration of iterations in the process which constitute a proxy for process quality.

Process optimization:

To compute Pareto-optimal processes, we did not consider optimization techniques which would aggregate the two optimization objectives – time and cost – into a single one (e.g. some monetary value). This decision was motivated by two major drawbacks of aggregation methods: 1) the requirement of problem specific knowledge as well as 2) the inability to determine portions of the entire Pareto front in one single optimization run. In section 3.2.3 we will discuss this issue more in detail.

Research gaps

With the scope of this work established we analyzed the resulting, still numerous, set of publications including strongly theoretical work as well as empirical field studies in order to identify crucial research gaps. As a result, four major gaps in literature concerning the time-cost optimization of PD processes could be revealed:

1. Assuming feedbacks in PD processes, neither the separate nor the combined behavior of crashing and overlapping in such a dynamic environment has been investigated yet. Crashing and overlapping constitute two managerial levers to accelerate PD processes and hence greatly impact time and cost.
2. The effects of *work policy*[8], a further important managerial lever for process cost and time, have not been researched yet, in particular its role on time-cost tradeoffs.
3. No work so far has investigated a "global" trade-off optimization of PD processes <u>simultaneously</u> considering a cyclic process structure and different work policies at a macro level (i.e. process level) as well as crashing and overlapping at a micro level (i.e. activity level).
4. It is noticeable that, in the literature, the calculation of process time and cost is strictly based on non-deterministic simulation methods rather than deterministic closed form analysis if the underlying process structure features iterative information flows.

Table 1.1 summarizes the current state of research in the context of this thesis. Essentially, we examined the managerial choices previous PD process models in the literature allowed, along with the proposed strategy to predict process cost and schedule (either via closed form

[8] See section 2.4 for more information. Basically, work policy refers to a set of rules governing cost and timing of activity execution as well as the allowed managerial levers.

Current State of Literature					
Process Model	**Time/Cost Calculation**		**Optimization**		**Exemplary References**
	Closed Form	Simulation	*Single-objective*	*Multi-objective*	
A	✓	✓	✗	✗	Malcolm et al. 1959 Kelley and Walker 1959
D	✗	✓	✓	✗	Browning and Eppinger 2002 Abdelsalam and Bao 2006
A/O	✓	✓	✓	✓	Roemer et al. 2000 Cho and Eppinger 2005
D/O	✗	(✓)	✗	✗	Cho and Eppinger 2005 (no explicit investigation of overlapping)
A/C	✓	✓	✓	✓	Kelley 1961 Fulkerson 1961
D/C	✗	✗	✗	✗	-
A/O, C	✓	✓	✓	✓	Roemer et al. 2004 Gerk and Qassim 2008
D/O, C	✗	✗	✗	✗	-

Table 1.1: Tabular literature survey.

analysis or via simulation) of the model and the optimization approach applied to it (either single-objective, multi-objective or none). As for the complexity of process models, we distinguished between 1) models merely permitting acyclic process architectures (denoted as "A") 2) models allowing iterations and thus dynamics in the process (denoted as "D") 3) models that incorporate a crashing policy (denoted as "C") and finally 4) models featuring an overlapping policy (denoted as "O"). Since the investigation of different work policies has not been considered in any model yet, we waived its explicit listing in Table 1.1.

Inherently, activity network based PD process models exhibit a process architecture either of type A_1 or A_2 whereas crashing and overlapping constitute variable options that can be added to an architecture. The resulting combinations for these four aspects of work policy are listed in the left most column of Table 1.1 yielding a total of 8x2x2=32 different scopes for publications to be surveyed. Note that each cell separately corresponds to a logical "AND" relationship between a model type and a mode for c/s calculation or optimization. If at least one publication could be associated to a certain combination, the corresponding cell was marked with a ✓ and vice versa it was marked with a ✗ if no related work has been published yet. Besides, we resigned on a complete reference list for each cell but instead quoted one or two exemplary references for each row in the table, if possible.

1.3 Thesis Objectives

Inspired by the aforementioned deficiencies we define the following high level objectives for this thesis. To some extent, the objectives build up on each other:

Thesis objective 1: Development of a richer PD process model accounting for feedback relationships in the process as well as for crashing- and overlapping policies. Additionally, different work policies with varying rules shall be developed.

Thesis objective 2: Development of a solution methodology for the newly developed process model in order to predict process cost and lead-time.

Thesis objective 3: Time-cost sensitivity analysis of major model parameters. Firstly, general scale-up behaviors for time and cost assuming cyclic processes shall be derived. Besides, the impact of work policy, crashing and overlapping in dynamic process environments shall be investigated. Based on the results of this study, new managerial insights shall be derived.

Thesis objective 4: Development of an efficient optimization framework capable of generating the time-cost Pareto-front of any PD process which applies the newly developed model and simulation.

Thesis objective 5: Multi-objective time-cost optimization of real-world PD processes which apply the developed model and simulation. Thereby, the insights gained through the use of artificial test processes shall be verified.

Research Gaps Covered In This Thesis				
Process Model	**Time/Cost Calculation**		**Optimization**	
	Closed Form	Simulation	*Single-objective*	*Multi-objective*
D	-	-	-	✓
D/O	-	✓	✓	✓
D/C	-	✓	-	✓
D/O, C	**-**	✓	✓	✓

Table 1.2: Research gaps covered in this thesis.

Thesis coverage

A summary of the objectives, presented at Table 1.2, clearly illustrates the research gaps which are supposed to be filled by the current study. The only interesting issue not covered in this work is an analytical solution for the time/cost calculation of iterative processes.

1.4 Thesis Organization

This thesis is essentially composed of four parts (Figure 1.2) and organized as follows. A first part comprising chapters 1-3 intends to foster the motivation and comprehension of the underlying thesis topic. Moreover, it provides background information necessary to understand the following chapters. For this purpose, chapter 2 introduces the most fundamental parameters of a PD process, subject to the purposes of this work, as well as their time-cost simulation. Subsequently, chapter 3 elaborates on the actual time-cost problematic for PD processes. Since the identification of the optimal time-cost tradeoffs constitutes a multi-objective optimization problem, the reader is additionally equipped with basic knowledge on multi-objective concepts and optimization techniques capable of computing the Pareto-front of the time-cost problem.

The second part of the thesis, including chapters 4 and 5, covers the new modeling and simulation of PD processes. Firstly, chapter 4 introduces a new, more realistic PD process

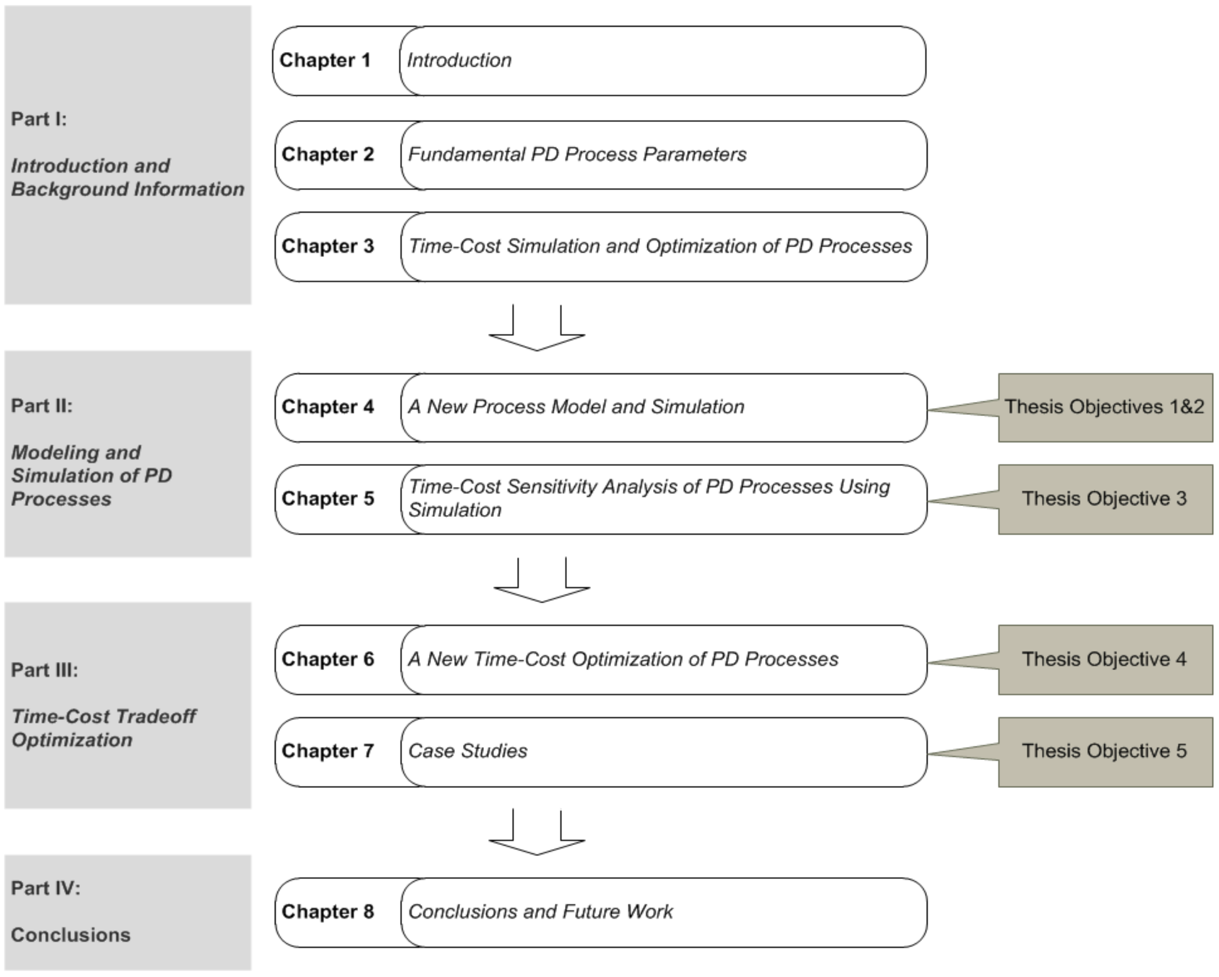

Part I:
Introduction and Background Information
Chapter 1
Introduction
Chapter 2
Fundamental PD Process Parameters
Chapter 3
Time-Cost Simulation and Optimization of PD Processes
Part II:
Modeling and Simulation of PD Processes
Chapter 4
A New Process Model and Simulation
Thesis Objectives 1&2
Chapter 5
Time-Cost Sensitivity Analysis of PD Processes Using Simulation
Thesis Objective 3
Part III:
Time-Cost Tradeoff Optimization
Chapter 6
A New Time-Cost Optimization of PD Processes
Thesis Objective 4
Chapter 7
Case Studies
Thesis Objective 5
Part IV:
Conclusions
Chapter 8
Conclusions and Future Work

Figure 1.2: Illustration of thesis outline.

model and presents how time and cost of processes applying this new model can be approximated via simulation. Then, chapter 5 concludes the second part by a thorough sensitivity analysis of the model. Thereby, 48 million artificial test processes were simulated to obtain general insights on the time-cost behavior of the most important process parameters in a cyclic process environment.

Next, chapters 6 & 7, representing the third thesis part, exclusively cover the time-cost tradeoff optimization. In chapter 6 we introduce an appropriate multi-objective optimization method for the novel process model in order to obtain Pareto-optimal process solutions. Subsequently, this optimization strategy is applied to two real-world processes in order to demonstrate the practical application of our new model, simulation and optimization. Moreover, previous insights based on artificially constructed processes are verified. Finally, this work ends with a short fourth part containing conclusions and an outlook for future research.

2. Fundamental PD Process Parameters

In order to accomplish the thesis objectives, it is essential to be aware of the fundamental PD process parameters which must be modeled, simulated and finally optimized. Thereby, we consider those process parameters as fundamental which 1) significantly affect its time and cost on one hand but 2) which can be also modified in practice by project managers on the other hand. Subject to the previously defined scope of the underlying work, we identified four of these parameters: process architecture, crashing- and overlapping policy as well as work policy. In the following we will briefly introduce the reader to each of them in a separate section.

2.1 Process Architecture

2.1.1 Introduction

Establishing an appropriate *process architecture* is critical for any successful PD strategy as it significantly affects project costs and lead-time. For example, Ahmadi et al. (2001) report on significant improvement in terms of PD lead-time and costs as a result of re-arranging the existing process architecture of a manufacturer for rocket engines. The process architecture defines not only the process activities but also the pattern of interaction among all activities which in turn impacts process cost and schedule:

Definition 2.1 *The process architecture describes the activities of a process and their dependency relationship as well as their sequence of execution* (Browning and Eppinger 2002)

Activity networks

In this work, we model process architectures as a network consisting of nodes and directed arcs thus prescribing a directed graph (digraph). This network-based view of a process can be generally classified into activity-on-arc (AoA) networks or activity-on-node (AoN) networks. In case of AoA networks, nodes represent events (typically start and finish dates of activities) in the process whereas arcs constitute activities. Contrary, AoN networks are probably more direct and frugal since nodes denote activities and arcs simply describe logical relationships, e.g. predecessor relationships, between them. Typically, both network types are visualized with flowcharts or traditional graph-based representation schemes.

Iteration and feedback

However, these formats are obviously not convenient to compare different architectural differences, in particular if numerous activities and relationships are involved (Browning and Eppinger 2002). Moreover, due to the importance of *iterations* in PD, proper network-based models have to account for feedback information flows and consequently cyclic process architectures. Iterations have been widely recognized as a fundamental characteristic of PD processes (e.g. Eppinger et al. 1994, Fricke 2000). Eppinger et al. (1997) defined iteration as the repetition of activities to enhance an evolving development process. In this work we define iteration as follows:

Definition 2.2 *Iteration represents the rework of an activity due to information feedback.*

Feedback occurs when a design activity commences based on missing (or uncertain) predecessor information if the (complete) predecessor information is received after the start of the activity (Figure 2.1). The late arrival of predecessor information could be due to detected errors, failure to meet requirements, or changing design directives, to name just a few of the common causes. There is no comprehensive classification scheme for iteration, although the literature refers to different types of iteration as planned versus unplanned, sequential versus parallel, good versus bad, and major versus minor (Safoutin and Smith 1998). Generally, iterations are admitted to enhance product quality (Whitney 1990, Safoutin and Smith 1998), while also being a driver of cost and lead-time. Therefore, it is clear that effective management of iteration is required to plan and control project cost, duration, technical performance (or quality), and risk.

Iteration can be managed in many ways including by: improving the sequence of design activities to streamline information flows, which form the basis for activity dependencies (Yassine and Braha 2003); developing new engineering automation tools to perform iterations

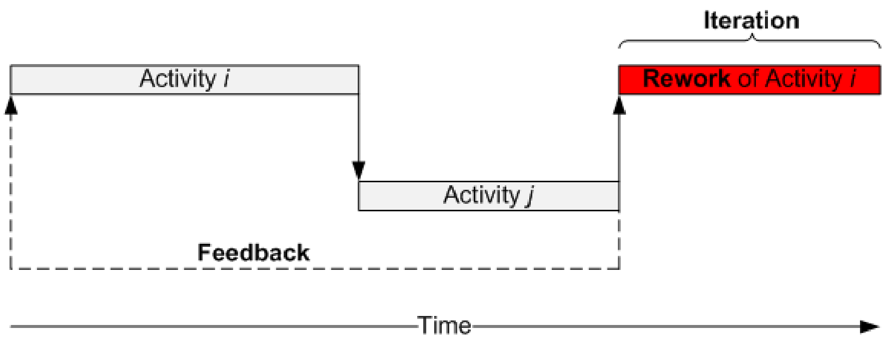


Figure 2.1: Demonstration of iteration.

faster (Yassine et al. 2000); adding resources at bottleneck activities (Yassine et al. 2003); and possibly limiting the scope of the development effort. Although activity iteration could be an intrinsic property of the process architecture, it is also a side effect of overlapping (see section 2.3) two nominally sequential activities.

Modeling activity networks with the Design Structure Matrix

Traditional AoA models (e.g. PERT) or AoN models do not allow to incorporate feedbacks. Therefore, numerous models have been constructed to allow feedback information flow within the process. Neumann and Steinhardt (1979) extended the PERT technique by probabilistic feedback loops and Adler (1995) developed a model based on queuing theory principles to consider iterations. Furthermore, Eppinger et al. (1997) used signal flow graphs to model all types of information flow while Ahmadi et al. (2001) applied markov chains. Though, these models rely on limiting assumptions, e.g. for the model used in Ahmadi et al. (2001) the sum of feedback probabilities for a certain activity must not exceed 1.0, and/or their format is neither concise nor suited to highlight iterations (e.g. Adler 1995, Eppinger 1997). Due to these shortcomings, which do not capture the real complexity of PD processes, and because of the amenability to matrix-based analysis we decided to utilize a digraph-based information flow model called the *design structure matrix* (DSM)[9], introduced by Steward (1981). Many DSM models with different objectives have been developed (e.g. to identify critical activities (Smith and Eppinger 1997)), but for our purposes so called *activity-based DSMs* are best suited as they represent the flow of activities over time which is particularly useful for highlighting iterations in the process (Browning 2001):

Definition 2.3 *A binary activity-based DSM is equivalent to the adjacency matrix of the underlying process digraph/AoN network. The n_A nodes of the digraph (representing the activities in a process) correspond to the n_A column and row headings in the matrix. The*

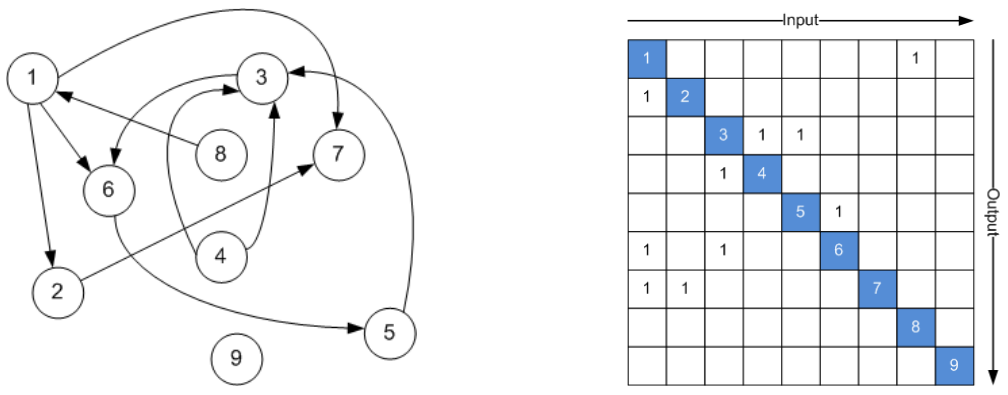
[9] The term *Dependency Structure Matrix* is also commonly used.

arrows, indicating a relationship between activities, correspond to the marks inside the matrix.

Hence, as shown in Figure 2.2, an activity-based DSM is a square matrix that shows activities along its diagonal, activity outputs in its columns and activity inputs in its rows.[10] As the number of activities and relationships grows, the matrix-based visualization provides significant advantages over digraphs and other models in visualization of feedback loops which are immediately obvious as super-diagonal marks. In order to model a process architecture using DSMs, the following three steps are required.

2.1.2 Definition of activities

The first step in establishing a process architecture is the definition of convenient activities capable of producing the process' required result of value. The project management literature typically proposes the use of a work breakdown structure (WBS) to identify high-level activities which are subsequently decomposed (Browning and Ramasesh 2007). In contrast, the PD literature provides a variety of generic, standard processes (like depicted in Figure 1.1) with default activities (e.g. Whitney 1990, Ulrich and Eppinger 2004). In fact, standardizing activities might be extremely useful for well-known processes in stable environments



(a) Digraph representation (b) DSM representation

Figure 2.2: A small process model visualized with a digraph (a) and an activity-based DSM (b).

[10] Some DSM literature uses the opposite convention (the transpose of the matrix), with inputs in columns and outputs in rows, and thus feedback below the diagonal; the two conventions convey equivalent information.

and a set of standard activities could serve as an initial activity list for new processes.

However, when choosing activities, process planners should also consider internal or external prescriptions and regulations (i.e. activities might be mandated in some contexts) as well as an activity’s ability to reduce risk and uncertainty (Browning and Eppinger 2002, MacCormack and Verganti 2003, Browning and Ramasesh 2007). Besides, process ambiguity and complexity often impede the activity definition in advance of process execution thus requiring an adaptive selection of activities based on their added value (Lévárdy and Browning 2009). Also, process planners have to think about the level of detail for process modeling, and consequently about the level of granularity for activity selection (Browning and Ramasesh 2007). Apparently, choosing the “right” activities of a process is a non-trivial task and involves many aspects to be considered.

2.1.3 Definition of activity dependencies

Like aforementioned, a process architecture does not only define a set of process activities but also their dependencies among each other. The literature distinguishes three dependency types between any pair of activities *i* and *j*: 1) *independent* activities which are simply not related at all 2) *dependent* activities, i.e. either activity *i* depends on *j* or vice versa *j* depends on *i* 3) *interdependent* activities which depend on each other. As a consequence of these activity dependencies, information can be exchanged in four different ways between an *upstream* activity *i* and a *downstream* activity *j* (Figure 2.3a). In case *i* and *j* are independent we note only concurrent information flows between them hence allowing the *parallel* execution of both activities. Contrary, dependent activities imply two possibilities for information flow: either *i* delivers information to *j* through a *sequential feed-forward relationship* or *j* provides output to *i* which constitutes a *sequential feedback relationship* hence potentially provoking the rework of *i*. Arguably the most interesting flows, namely *coupled information flows*, are caused by interdependent activities: since *i* requires the output information of *j* as input and *j*

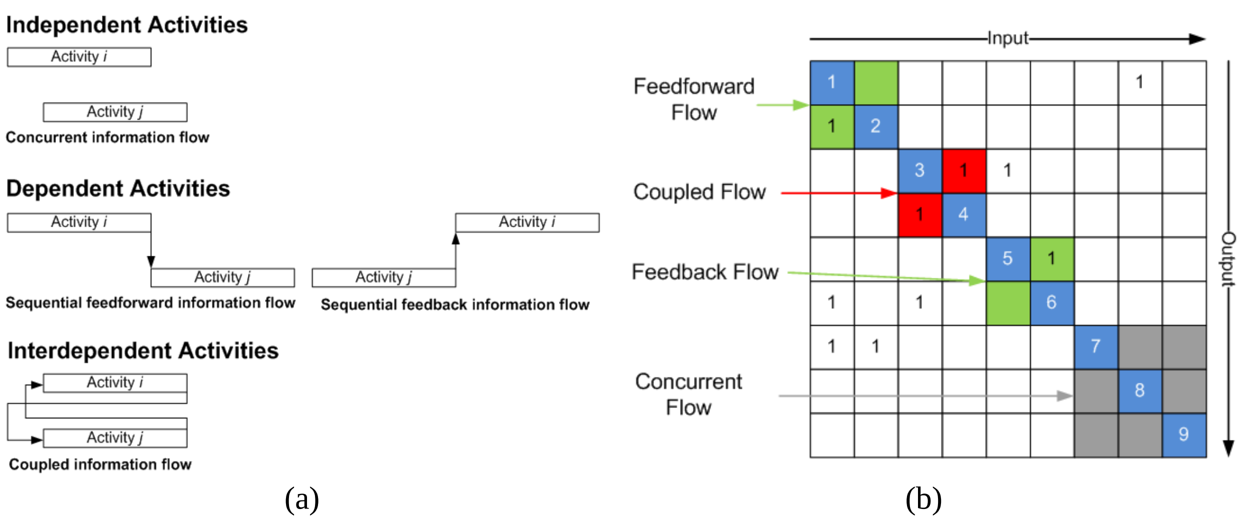


Figure 2.3: Information flows between two activities (a) and within an activity-based DSM (b).

needs the output of *i* as input, we realize a “chicken and egg” problem. In order to solve it, the upstream activity *i* has to carefully make assumptions about the output of downstream activity *j* as wrong or imprecise assumptions could lead to an undesirable output of *i* thereby increasing the probability of unintentional activity iterations later in the process. Therefore, several possibilities have been suggested to reduce the number of coupled information flows (Browning 2001), including 1) aggregation of two coupled activities to one 2) decomposition of coupled activities or 3) applying a technique called *Tearing* (Steward 1981).

With the different information flows between activities established, we note that feed-forward flows in the process are shown below the diagonal of activity-based DSMs, and non-zero super-diagonal cells indicate the presence of feedback information flow (Figure 2.3b). While any concurrent activities *i* and *j* are intuitively modeled through an empty mark (or a zero value) in cells $m_{1,ij}$ and $m_{1,ji}$ of an activity-based DSM M_1, coupled information flow among *i* and *j* is indicated by non-zero values in both corresponding matrix cells.

2.1.4 *Definition of the activity sequence*

So far, process activities have been selected and their dependencies among each other have been determined. Though, one crucial component of any process architecture is still missing:

the *activity sequence*. We refer to this term as the order of activities in the rows or columns of an activity-based DSM indicating the sequence of activity execution. Altering the activity sequence of any process architecture, subject to possible predecessor constraints (e.g. due to technical feasibility issues), may change the information flow among its activities thus resulting in a novel architecture. Though, re-sequencing the activities does neither add new relationships to the process nor does it remove existing ones. Hence, it does not change activity dependencies at all. Instead, it possibly converts sequential feed-forward flows to feedback flows or vice versa thereby greatly influencing the extent of iteration in the process. This transformation can be also regarded as dilemma between "waiting for inputs" (i.e. sequential feed-forward) and "making assumptions about the input" (i.e. sequential feedback). Consequently, sequencing of activity may commute prerequisites to assumptions. In contrast, coupled or concurrent information flows cannot be affected by varied activity sequences.

Let us briefly demonstrate the described effect of activity sequencing with a simple example using the process displayed in Figure 2.2. By exchanging the position of activities 1 and 7 in the original DSM (Figure 2.4a)[11], we define a new activity sequence and consequently obtain a new process architecture, depicted at Figure 2.4b. The visual and concise format of the corresponding DSMs quickly illuminates their structural differences.

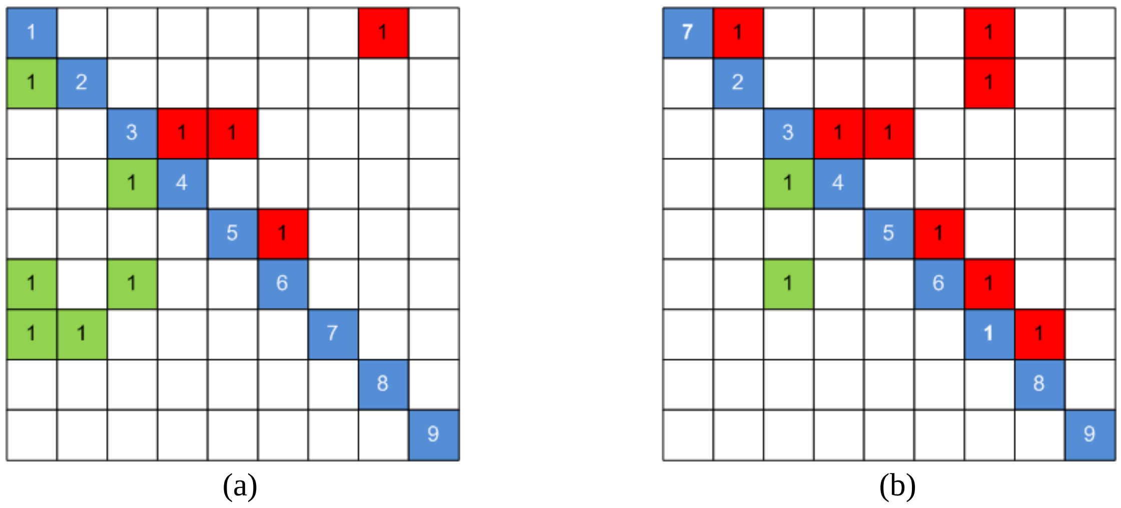


Figure 2.4: Original DSM sequence (a) and re-sequenced DSM with different architecture (b).

[11] Assuming that no constraints prohibit this exchange.

While the original activity sequence results in an architecture which primarily features feed-forward information flow (highlighted in green color) and only a moderate number of feedbacks (highlighted in red color), the new process architecture is characterized by much more feedback marks which are supposed to augment the extent of iteration and rework. Noteworthy, all feed-forward relationships associated with the interchanged activities 1 and 7 have been converted and appear as feedbacks in the re-sequenced process.

It is clear that the activity sequence's impact on process cost, schedule and uncertainty may be enormous. But due to the factorial growing number of possible activity sequences with increasing number of activities n_A (i.e. $n_A!$) it is mostly too time consuming to evaluate every sequence according to some metric (e.g. cost) and to choose the best one, even for moderate n_A. In lieu of randomly selecting any activity sequence, researches have proposed a variety of deterministic sequencing objectives for DSM modeled processes (see Meier et al. 2007 for an overview) which can be used as objective functions for search algorithms. Most of them focus on minimizing super-diagonal marks in the DSM, i.e. the number of feedback relationships. The motivation is not only an expected reduction of cost and lead-time due to less iteration/rework but also a reduction of process uncertainty since feedbacks represent assumptions instead of actual information. However, all of these proposed objectives are merely proxies for the actual goals of decreasing the process's overall duration, cost, and risk.

2.2 Crashing

Essentially, crashing boils down to an attempt to get the most effectiveness out of the least time and expense:

Definition 2.4 *<u>Crashing</u> is a popular approach to compress process lead-time through the assignment of additional resources (e.g., overtime, more or faster machinery, software tools) to process activities. Importantly, the new, shorter process retains the original information flow.*

Although it is not necessarily a special feature for scheduling PD projects, however, we include a crashing feature in our proposed model in chapter 4 since crashing of activities has been widely used in process management to reduce project lead-time (Kerzner 2005). The immediate side effect of this strategy is typically a raised budget as most resources will not be for free. Though, higher initial cost are accepted in certain competitive circumstances due to the anticipation of greater long term market shares and product profitability resulting from an earlier product launch. Also, if a PD process is behind its schedule, managers are often disposed to bring the project back "on track" at the expense of a greater budget. Consequently, a tradeoff situation between process lead-time and process cost arises, making the definition of a proper crashing strategy for an entire process to a nontrivial task. Indeed, crashing constitutes a powerful mean to reduce process duration but it has to be carefully planned and executed in practice to unfold its positive effects.

In fact, crashing may shorten an activity's duration, thus decreasing overall project duration if the activity is on the critical path.[12] Though, it is important to mention that previous literature modeled crashing merely as time decreasing and cost increasing although practice shows that, despite the allocation of additional resources, duration is sometimes also extended (Brooks 1995) - especially if we consider human resources. New staff must not necessarily be familiar with the tasks at hand and may be less productive than already assigned staff. Moreover, new human resources must be taught and guided by usually the most experienced members of the PD process who could work themselves to get the activity finished. As a consequence, work progress of a crashed activity could be slowed down and unintentionally elongate overall process lead-time.

2.3 Overlapping

In contrast to crashing, influencing the process with intra-activity changes, overlapping

[12] Notably, the critical path may change as a consequence of crashing.

constitutes an approach which considers inter-activity relationships:

Definition 2.5 *<u>Overlapping</u> of development activities is one of the basic pillars of Concurrent Engineering (CE) practices* (Funk 1997, Smith 1997) *considering downstream concerns in upstream phases as opposed to a strictly sequential stage-gate workflow between activities* (Wang and Yan 2005, Yassine and Braha 2003).

The basic overlapping model deals with overlapping two sequential activities in a PD project, referred to as an *upstream* feeding activity *i* and a *downstream* dependent activity *j* (see Figure 2.5a). Most of the literature on overlapping applies this simple two activity scenario rather than modeling overlapping of multiple activities (probably for reasons of complexity). However, in chapter 4 we will introduce a process model which accounts for multiple activity overlapping. The main objective of a basic overlapping problem is to find the best overlapping magnitude between *i* and *j* that minimizes total lead-time (see Figure 2.5b). Although overlapping *i* and *j* gives the downstream activity *i* a head-start, the benefits of activity overlapping might be partly eliminated since potential deficiencies exist as well (Krishnan et al. 1997a, Wang and Yan 2005).

As downstream activities are allowed to start without complete input information, additional uncertainty is introduced into the process: the crucial information provided by upstream activities might be received either too late or in poor quality. Therefore, unintentional iteration in the process might occur requiring downstream rework which could boost a process's overall cost and duration (see Figure 2.5c). Additionally, forcing upstream activities to produce information early in order to pass it to downstream activities could result in a quality loss for upstream activities (Krishnan et al. 1997a). Mansfield et al. (1979) and Eastman (1980) both provided empirical evidence for PD processes with increased costs due

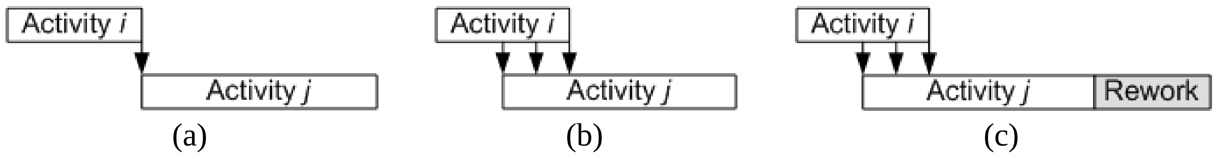


Figure 2.5: Illustration of overlapping two activities.

to overlapping stages. However, MacCormack et al. (2001) argued that overlapped PD process stages enhance flexibility, which in turn may save money in uncertain and dynamic environments.

Due to this tradeoff (process speed vs. rework), the success of overlapping requires accurate management of interaction and communication (Terwisch and Loch 1999, Ford and Sterman 2003). Potential candidates for overlapping have to be selected carefully along with a reasonable intensity of overlapping (AitSahlia et al. 1995). Evaluating the effects of overlapping in an industry-specific study, Loch and Terwisch (2005) confirmed the time savings of overlapping in general while arguing that its specific impact depends on a firm's capability to resolve uncertainty early in the process.

2.4 Work Policy

Often neglected in past studies, work policy may have a substantial impact on process measures, such as cost, schedule or uncertainty.

Definition 2.6 *The term* *work policy*, *introduced by* Browning and Eppinger (2002), *refers to the set of rules governing the timing of activity execution. Moreover it outlines the managerial levers available (e.g. crashing, overlapping) and which choices are possible. Consequently, the work policy affects not only timing but also cost and eligibility.*

Accordingly, the work policy may have a tremendous impact on process cost & duration and eventually project success. Therefore its definition is mostly a task for senior process/project managers. Essentially, a work policy in the context of this thesis addresses the following fundamental questions:

- Which activities may work concurrently?
- Which activities are allowed to be crashed?
- Which activities may proceed without complete input information?
- Which activities can make assumptions about its inputs?

- How is iteration managed?
- When do activities work off their rework?

3. Time-Cost Simulation and Optimization of PD Processes

This chapter firstly provides background information on the simulation of the process parameters, introduced in the precedent chapter, with respect to time and cost. Subsequently, the second section introduces the classical time-cost tradeoff problem in PD and furthermore exposes research gaps in this context. Finally, this chapter concludes with an introduction of optimization methods capable of solving multi-objective problems, such as the time-cost problems.

3.1 Time-Cost simulation of PD processes

While traditional acyclic network models (CPM, PERT) allow for a deterministic calculation of cost and schedule, cyclic information flows subject to stochastic feedback probabilities dramatically complicate a similar closed form analysis. The author is not aware of any publication to date (see Table 1.1) which presented a deterministic strategy to calculate duration and cost for cyclic process models. As the use of simulation is widespread for stochastic processes of any kind, numerous PD process simulation models have been published in the PD literature. We like to resign on the enumeration of all simulation models but note that the core difference lies in two aspects: 1) the modeling of the dynamic progress of the process and 2) the sampling of activity cost and duration from known probability density functions (PDFs).

Having selected DSMs for the modeling of the process, we note that several DSM-based simulation models (and tools) already exist in the literature (Browning and Eppinger 2002, Cho and Eppinger 2005, Abdelsalam and Bao 2006). While all these simulations are mainly aimed at determining the process completion time and cost for a given task arrangement, some exhibit an additional component for determining the optimal architecture as well (e.g. Abdelsalam and Bao 2006) by testing (i.e. calculating time and cost) various architectural arrangements. However, if we focus on the simulation (and not optimization) component, then minor differences would only exist between the various PD process simulation models.

Aside from the detailed differences that are inconsequential to this thesis, all DSM-based simulation models work as follows. First, as input, a DSM simulation model usually requires a binary DSM based process model and some additional data. For each activity interface (i.e. marks in the DSM), the model requires an assessment of the probability of a typical change in the data causing rework for a dependent activity as well as the impact and the learning curve effect of that rework should it occur. Activity duration and cost are random variables, represented by *probability density function* (PDF), e.g. a triangular PDF using three point estimates: best duration, likely duration, and worst duration.

Then, the simulation quantifies a process configuration's expected cost, duration and variance. Variances in duration and cost are largely attributed to the number of iterations required in the process and their scope. Since task rework may or may not occur (depending on a probabilistic check of feedback relationships in the process), the simulation model treats iterations stochastically, with a probability of occurrence depending on the particular package of information triggering rework.

3.2 Potential Time-Cost Tradeoffs for PD Processes

The classical time-cost tradeoff problem

By assuming acyclic process architectures, previous research was capable to analytically calculate/approximate the impact of changes in the crashing- and overlapping policy on cost and duration. Overlapping and crashing are both means applied in practice to reduce PD time, but they usually subject the process to additional cost as pointed out in section 2.2 and 2.3. Even if speed-to-market is decisive for successful PD, it is clear that companies are not willing to invest an unlimited amount of money into their projects. Firms often have a predefined budget for a PD project which must not to be exceeded. A *time-cost tradeoff* situation clearly arises as organizations seek the fastest PD process through managerial decisions subject to a predefined budget, or vice-versa, identifying the cheapest PD process for a given project duration.

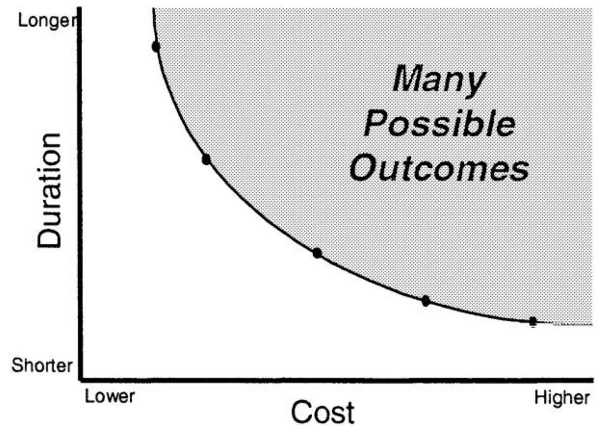
The time-cost tradeoff problem for PD processes has been a long-term subject of research. Interestingly, in the early years of traditional project management, the time-cost tradeoff was treated solely as a crashing problem. Kelley and Walker (1959) were the first to state the problem of computing the least-cost curve for a project composed of activities, which have an associated normal completion time and a crashed completion time, as a parametric linear problem. Then, Kelley (1961) and Fulkerson (1961) independently proposed two different efficient algorithms based on network flow methods to solve this problem with a linear cost function. Later publications investigated more complex cost functions exhibiting convexity (Berman 1964), concavity (Falk and Horowitz 1972) or assigning penalties to pivotal activities (Kanda and Rao 1984).

However, most activity network-based research on time-cost tradeoffs assume an acyclic network, as compared to cyclic networks that are noted in PD project scheduling environments. Recently, Roemer et al. (2000) also analyzed the time-cost tradeoff due to overlapping and bridged the gap between the literature on overlapping and crashing by demonstrating the benefits of jointly applying both strategies in a subsequent publication (Roemer and Ahmadi 2004).

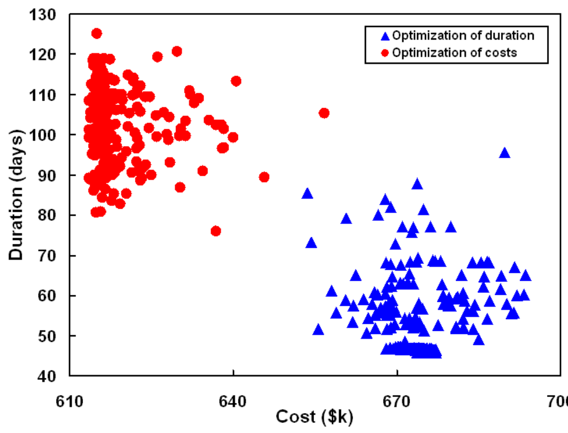
Time-Cost tradeoffs due to changes in the process architecture

While crashing and overlapping have been widely recognized for their role on time-cost tradeoffs, we note that tradeoffs due to changes in process architecture or because of work policy rules have been mostly neglected so far. Advanced complexity of analytical calculations for cyclic process architectures might be the reason for this research deficiency. Though, simulating PD processes, as pointed out in the previous section, allows us to approximate the impact on cost and duration through changes in the architecture or work policy.

In the pioneering work of Browning and Eppinger (2002), the authors proposed a cyclic process model using a discrete event simulation in order to show that pure manipulations of the process sequence may lead to different time-cost outcomes. In fact, both authors expected a time-cost tradeoff exclusively emerging by a rearrangement of the activity sequence (Figure 3.1). This hypothesis could be confirmed in a study by Meier (2006) who separately optimized the activity sequence of the identical PD process in the aerospace industry with respect to cost and duration using a simulation as objective function.



(a) Adapted from Browning and Eppinger (2002).



(b) Adapted from Meier (2006).

Figure 3.1: Time-Cost tradeoffs through changes in the process architecture.

Figure 3.1b illustrates the corresponding results of the optimization thereby clearly indicating the different regions of the search space for the optimal process duration and process cost.

Iterative overlapping

Both studies of Browning (2002) and Meier (2006) demonstrated that process sequences with low values for lead-time exhibited in return high cost, and vice versa such process sequences with long durations featured low cost. Thereby, an explicit reduction of process lead-time could be established by favoring highly iterative processes over sequential ones. This important concept is called *iterative overlapping* (Browning and Eppinger 2002; Krishnan et al. 1997b):

Definition 3.1 *Iterative overlapping refers to a strategy which allows an activity to be started concurrently with its predecessors, instead of waiting for final results of all its predecessors, in order to allow designers to set up and "get their feet wet"* (Browning and Eppinger 2002).

The benefits of iterative overlapping are typically achieved through low rework impact values and high learning curve effects of activities on the critical path which can be started in parallel with other activities while no complete input information is available (Figure 3.2). This strategy clearly increments the probability of rework, in particular if the output of the upstream activity is difficult to forecast. But in case of low rework impact values and high

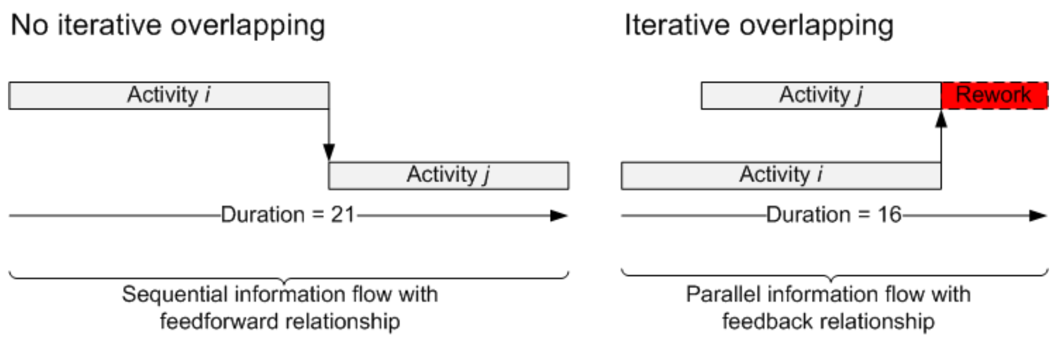


Figure 3.2: Illustration of iterative overlapping.

learning curve effects overall lead-time can still be shortened. On the other hand, the downside of iterative overlapping constitutes a rise of the process' budget due to rework as well as an increased uncertainty within the process and higher coordination effort.

3.3 Time-Cost Optimization

3.2.1 Introduction

Introduction

When dealing with time-cost tradeoffs, the objective is to identify the set of all best tradeoff solutions of the problem – the *Pareto-front*. Thus, we have to apply *multi-objective optimization* (MOO) methods rather than single-objective ones in order to determine this set. Fortunately, much attention has been paid in the last few years to the optimization of problems whose formulation involves simultaneously optimizing multiple, contradicting, objective functions. This interest is mainly motivated by the multi-objective nature of most real world problems (Deb 2009). Contrary to *single-objective optimization problems* (SOOPs), a single best solution with respect to all objectives usually does not exist for a *multi-objective optimization problem* (MOOP). Instead, one seeks a set of the best tradeoffs, which consists of elements that are superior to others in at least one of the objectives (in this thesis: time or cost) while inferior in the remaining objectives. This set is commonly known as *Pareto-optimal* set and its solutions are called *non-dominated* solutions which lie in a region of the search space called the Pareto frontier, or Pareto-front for short. The remaining solutions in the search space are called *dominated* solutions.

Exemplary multi-objective optimization problems

Let us examine some examples for MOOPs. Fujita et al. (1998) optimized a four-cylinder automotive gasoline engine with respect to fuel consumption, acceleration performance and starting response (Figure 3.3a). In Poloni et al. (2000), Poloni applied MOO methods in order to seek a trade-off among hydrodynamic lift and hydrodynamic drag for a sailing yacht fin keel (Figure 3.3b). Another interesting application refers to the optimization of low-thrust spacecraft trajectories from Earth to Mars and from Earth to Mercury (Coverstone-Carroll et

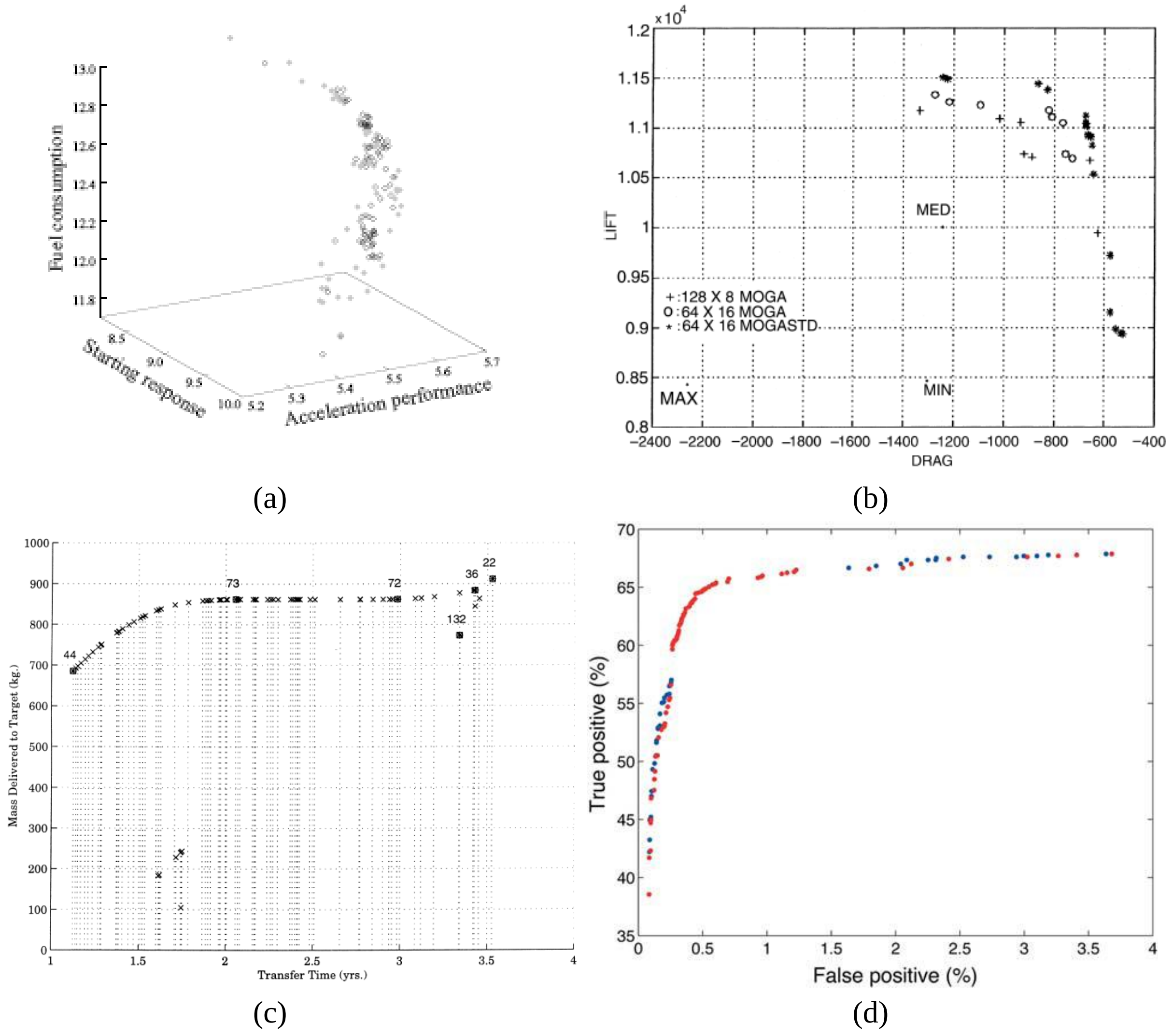


Figure 3.3: Illustration of Pareto-fronts for different real world applications, adapted from Fujita et al. (1998) (Figure 3.3a), adapted from Poloni et al. (2000) (Figure 3.3b), adapted from Coverstone-Carroll et al. (2000) (Figure 3.3c), adapted from Everson and Fieldsand (2006) (Figure 3.3d).

al. 2000) in order to minimize travel time while maximizing payload mass (Figure 3.3c). Finally, Everson and Fieldsend (2006) cast the tuning of safety related systems like the Short in the number of false alerts (false positive) served as two contradicting objectives (Figure 3.3d). The literature on multi-objective real-world applications is abundant with many more examples and the reader may refer to (Coello et al. 2007, Deb 2009).

3.2.2 Multi-objective definitions

Definition of an MOOP

After the informal introduction we formally define basic multi-objective concepts, mainly used in Part III of this work, starting with the definition of an MOOP:

Definition 3.2 *A typical multi-objective optimization problem (MOOP) with m decision variables and n objectives can be formulated as:*

$$\begin{aligned} \text{min/max}\ \ & \boldsymbol{Y} = f(\boldsymbol{X}) = (f_1(\boldsymbol{X}), \dots, f_n(\boldsymbol{X})) \\ \text{where}\ \ & \boldsymbol{X} = (x_1, \dots, x_m) \in \boldsymbol{X}^* \\ & \boldsymbol{Y} = (y_1, \dots, y_n) \in \boldsymbol{Y}^* \end{aligned} \tag{3.1}$$

with **Y** *representing the objective vector,* **X** *denotes the decision vector,* $\boldsymbol{X}^*$ *constitutes the decision space and* $\boldsymbol{Y}^*$ *the objective space.*

Concrete examples for $\boldsymbol{X}^*$ in the context of this thesis comprise the crashing intensity for a certain activity or the overlapping duration between two activities as well as the process architecture. Contrary, process cost and process duration can be regarded as examples for $\boldsymbol{Y}^*$. Constraints in the decision variables $x_1,\dots, x_m$ define the feasible region $\boldsymbol{X}^*$, and any point $\boldsymbol{X}\in\boldsymbol{X}^*$ defines a feasible solution. Basically, a function $f(\boldsymbol{X})$ maps decision variables in **X** to the corresponding objective values in **Y**. Figure 3.4 depicts such a mapping assuming the overlapping duration of an activity *i* as well as its crashing intensity as decision variables while activity cost and activity duration denote objective values. Please note that an extended

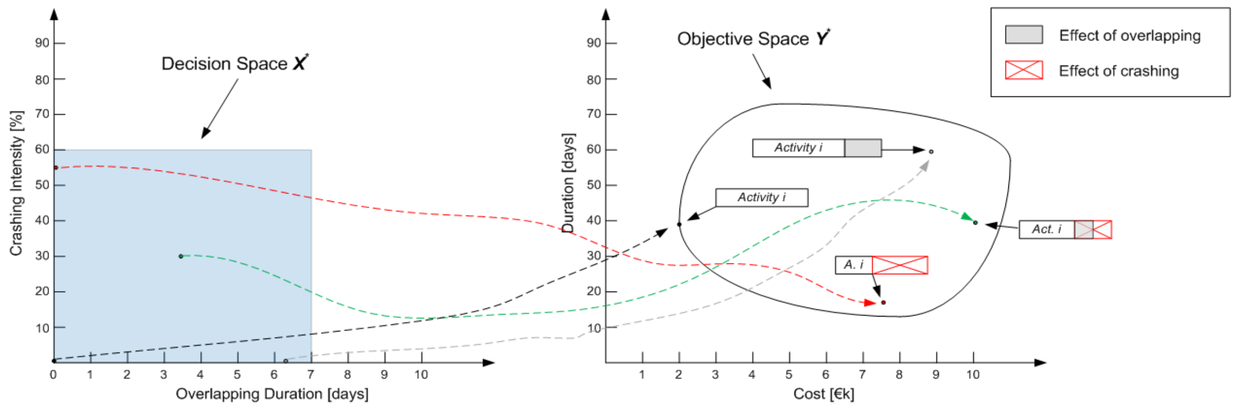


Figure 3.4: Mapping of decision variables to objective values.

overlapping duration of an activity may also elongate the entire activity duration due to rework as pointed out in section 2.3.

Domination and non-domination

Definition 3.3 *A decision vector* $\boldsymbol{A} \in \boldsymbol{X}^*$ *is said to* <u>*dominate*</u> *a decision vector* $\boldsymbol{B} \in \boldsymbol{X}^*$, *denoted as* $\boldsymbol{A} \prec \boldsymbol{B}$, *for a multi-objective minimization problem if the following two equations hold:*

$$\forall i \in \{1, \ldots, n\}: f_i(\boldsymbol{A}) \leq f_i(\boldsymbol{B}) \tag{3.2}$$

$$\exists j \in \{1, \ldots, n\}: f_j(\boldsymbol{A}) < f_j(\boldsymbol{B}) \tag{3.3}$$

Based on the above equations, Zitzler et al. (2000) defined non-domination and Pareto-optimality as follows:

Definition 3.4 *A decision vector* $\boldsymbol{A} \in \boldsymbol{X}^*$ *is* <u>*non-dominant*</u> *related to a set* $\boldsymbol{X}^+ \subseteq \boldsymbol{X}^*$ *if*:

$$\neg \exists \boldsymbol{B} \in \boldsymbol{X}^+: \boldsymbol{B} \prec \boldsymbol{A} \tag{3.4}$$

The decision vector $\boldsymbol{A} \in \boldsymbol{X}^*$ can be called *Pareto-optimal* if $\boldsymbol{A}$ is non-dominated relative to $\boldsymbol{X}^*$.

Pareto-front

The introduced terms ultimately allow us to formally define the Pareto-front:

Definition 3.5 *The* <u>*Pareto-front*</u> $\boldsymbol{F}^* \subseteq \boldsymbol{X}^*$ *can be defined as*:

$$\boldsymbol{F}^* = \{\forall \boldsymbol{A} \in \boldsymbol{X}^* \mid \neg\exists \boldsymbol{B} \in \boldsymbol{X}^* : \boldsymbol{B} \prec \boldsymbol{A}\} \tag{4.5}$$

Figure 3.5 provides a visual example. The two objectives y_1 and y_2 must be minimized. The curve on the lower-left of the shaded area indicates the Pareto-front. Points B and C are placed on the Pareto-front and cannot be dominated by any other feasible point while point A is dominated by many of the points (such as B) on the Pareto-front.

ε-dominance

Another highly-beneficial, multi-objective, concept regards ε-dominance (Laumanns et al. 2002, Helbig and Pateva 1994):

Definition 3.6 *A decision vector $\boldsymbol{A} \in \boldsymbol{X}^*$ is said to ε-dominate another decision vector $\boldsymbol{B} \in \boldsymbol{X}^*$ (written as $\boldsymbol{A} \prec_\varepsilon \boldsymbol{B}$) for an $\varepsilon > 0$ if the following equation holds*:

$$\forall i \in \{1, \ldots, n\}: (1 + \varepsilon) \cdot f_i(\boldsymbol{A}) \leq f_i(\boldsymbol{B}) \tag{3.6}$$

Applying the ε-dominance concept to the Pareto-front yields the following definition:

Definition 3.7 *A set of vectors $\boldsymbol{F}_\varepsilon^*$ is called the ε-Pareto set if the following two conditions hold:*

$$\boldsymbol{F}_\varepsilon^* = \{\forall \boldsymbol{B} \in \boldsymbol{X}^* \mid \exists \boldsymbol{A} \in \boldsymbol{F}_\varepsilon^* : \boldsymbol{A} \prec_\varepsilon \boldsymbol{B}\} \tag{3.7}$$

$$\boldsymbol{F}_\varepsilon^* \subseteq \boldsymbol{F}^* \tag{3.8}$$

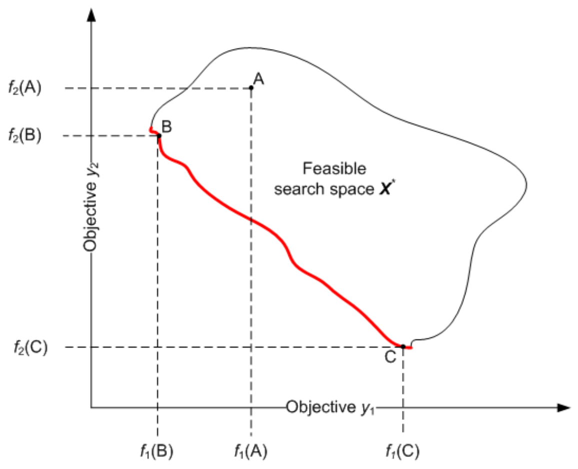


Figure 3.5: Illustration of Pareto-optimal solutions (points B and C) in a search space.

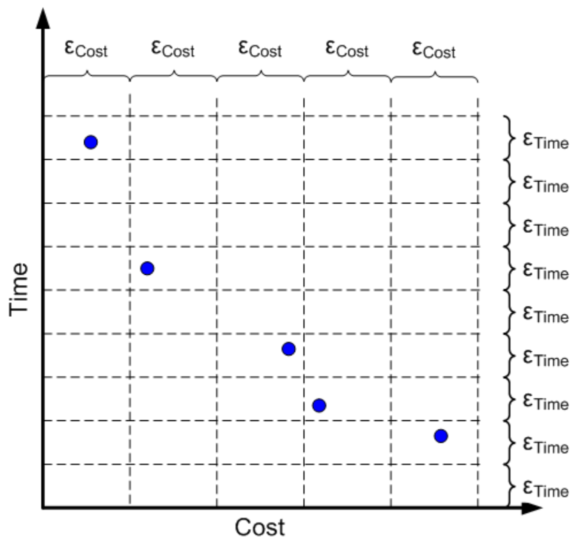


Figure 3.6: Illustration of ε-dominance.

The first condition (equation 3.7) ensures that every $\boldsymbol{F}_{\varepsilon}^{*}$ constitutes a so-called *ε-approximate set* of $\boldsymbol{F}^{*}$ (Helbig and Pateva 1994), and the second equation postulates that every ε-Pareto set $\boldsymbol{F}_{\varepsilon}^{*}$ contains Pareto solutions of $\boldsymbol{F}^{*}$ only. Basically, the ε-dominance reduces the cardinality of the Pareto-region by decomposing the objective space into multiple hyper-boxes. Figure 3.6 visualizes ε-dominance for two objectives (cost and duration).

3.2.3 Genetic algorithms for multi-objective optimization

Classification of MOO techniques

For MOOPs, such as the underlying time-cost tradeoff problem, numerous optimization strategies have been developed. Generally, MOO techniques can be classified into three major categories (Miettinen 1999, Collette 2003, Coello 2007):

1. *A priori techniques* including MOOP solver which assume a desired goal or a pre-ordering of the objectives performed by the decision maker prior to the search.
2. *Posteriori techniques* not requiring preference information from the decision maker prior or during the search.
3. *Progressive techniques* requiring input of the decision maker during the search to guide the search towards areas in the search space which perform the tradeoff between

objective functions that the decision maker would like to perform.

Figure 3.7 depicts all three categories with exemplary optimization methods for each class. A more detailed hierarchy of MOP methods is presented in Collette and Siarry (2003).

Multi-objective meta-heuristics

Interestingly, all of the optimization methods in Figure 3.7, except Meta-heuristics, convert the MOOP to a SOOP which subsequently has to be optimized by a single-objective optimizer (Deb 2009). Importantly, the obtained solution is specific to the parameters used in the conversion method and not guaranteed to be Pareto-optimal. Furthermore, aggregation methods, while easy to implement and converging to the Pareto front, are incapable of producing certain portions of the Pareto front in one single run but deliver a single solution instead.[13] Besides, they require additional problem-specific knowledge about the relative weightings of the various objectives. Due to these shortcomings of "classical"[14] aggregation

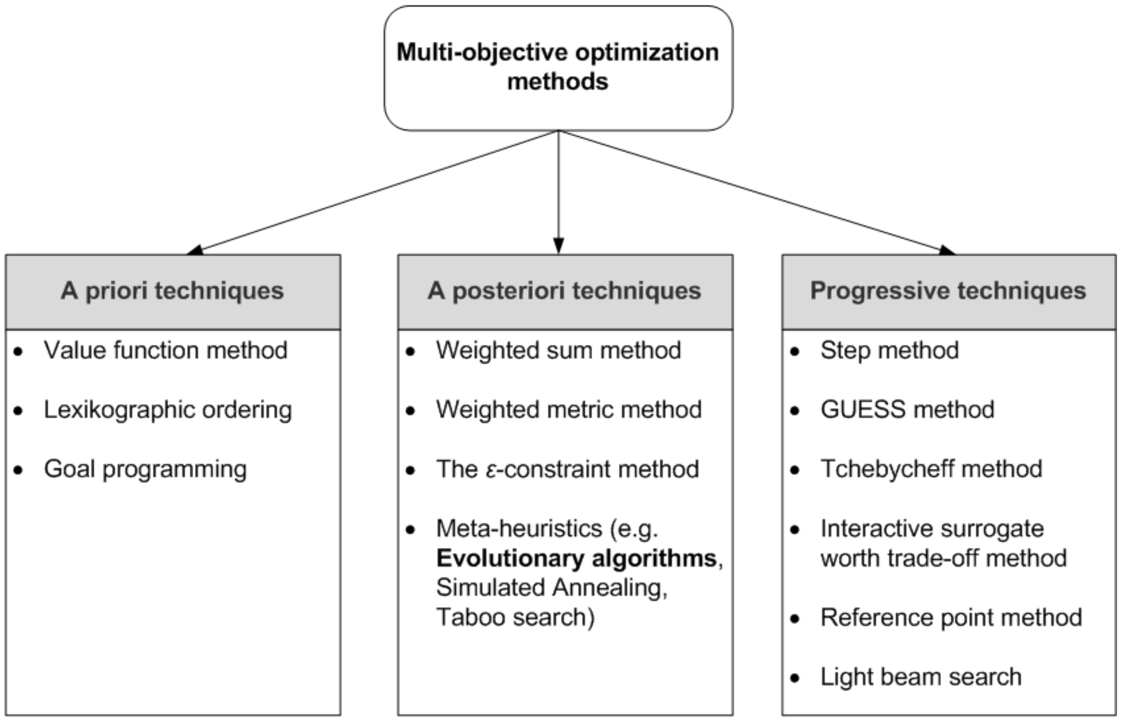


Figure 3.7: Classification of multi-objective optimization methods.

[13] For many aggregation methods we can prove that every Pareto-optimal solution corresponds to an optimal solution of the single-objective optimization problem.

[14] Deb (2009) applies the term "classical method" for search and optimization algorithms which use single solution updates and deterministic transition rules.

methods, multi-objective meta-heuristics have become very popular to solve real-world MOOPs.

Definition 3.8 *Meta-heuristics are methods that orchestrate an interaction between local improvement strategies and higher level procedures to create a process capable of escaping from local optima and performing a robust search* (Glover 2003).

Genetic Algorithms

Among a variety of meta-heuristics in literature, Genetic Algorithms (GAs) have received most attraction in literature.

Definition 3.9 *Genetic Algorithms (GAs) are search algorithms based on the mechanics of natural selection and genetics* (Goldberg, 1989).

Originally developed to solve SOOPs, GAs have been extended by multi-objective operators as their massive parallel search qualifies them to handle MOOPs extraordinarily well since the entire Pareto set (rather than a single Pareto point) can be quickly approximated in a single run. Due to these benefits, we developed in chapter 6 a MOGA tailored to our time-cost tradeoff problem.

Detailed information on Genetic Algorithms can be found in Holland (1975) or Goldberg (1989). However, other meta-heuristics like Ant Algorithms (Doerner 2004, Iredi 2001), Evolution Strategies, Particle Swarm algorithms (Coello et al. 2004) have been also enriched by a multi-objective version. It is important to note that a general superiority of one approach over the other cannot be claimed due to the *No Free Lunch theorems* (Wolpert and Macready 1997).

Simple GA flow

On a high-level perspective, GAs can be classified in *GA components* and *GA mechanics*. The components of a GA consist of 1) a proper data structure, called *chromosome* (by analogy to natural evolution), representing one point in the search space, as well as of 2) a *fitness function*. This function evaluates chromosomes in the objective space $\boldsymbol{Y}^*$ and subsequently assigns them a scalar value, called *fitness value*, which reflects the quality of the chromosome. Since GAs start their search from multiple points in $\boldsymbol{X}^*$, an entire set of chromosomes, called *population*, is maintained. Notably, a chromosome is not equal to a decision vector $\boldsymbol{X}$ but encodes it with an appropriate representation (e.g. a binary or integer representation).

In order to iteratively guide the search of the chromosomes, GAs incorporate special mechanics, namely *initialization*, *selection*, *crossover* and *mutation*. At the beginning of every GA, the initialization phase randomly generates a population according to the underlying encoding strategy for a chromosome. Afterwards, fitness values are assigned to every chromosome and a so called *mating pool* is created. This mating pool is filled by randomly sampling from the population using a proportionate- or ordinal based selection scheme. Basically, selection schemes are supposed to filter those chromosomes out of a population which are most promising for further search operations. Following selection, crossover and mutation are applied to the mating pool. While the crossover stage is responsible for a global exploration of the search space by combining significantly large traits of two chromosomes, mutation operators typically manipulate only small parts of a chromosome hence being often regarded as a local search mechanic. Subsequent to mutation, an entire iteration of a GA, denoted as *generation*, has been completed and it is checked whether a predefined convergence criterion (e.g. the number of generations) has been met. Figure 3.8 depicts the previously described simple flow of a GA. Not every GA design relies on the previously described mechanics and does not necessarily exhibit a search flow as depicted in Figure 3.8. Instead, additional, and often more complex mechanics can be found in literature. Though, the aforementioned mechanics can be regarded as "backbone" for every GA/MOGA as they appear in every GA/MOGA to some extent.

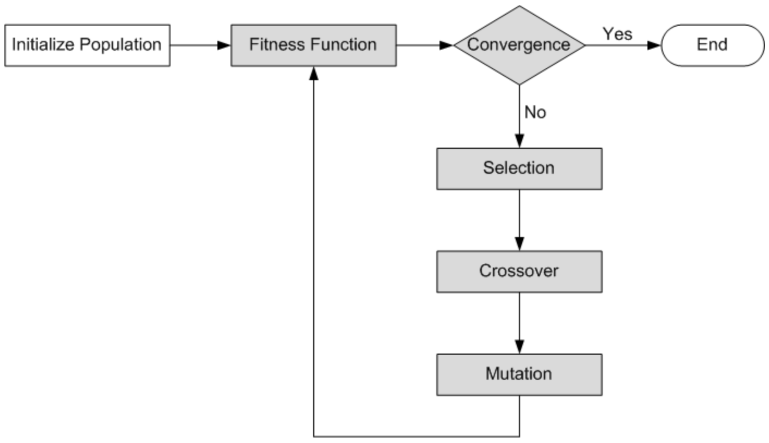


Figure 3.8: Illustration of a simple GA flow.

Multi-objective GAs

Multi-objective GAs retain many of the single-objective GA features. Though, by contrast to single-objective GAs, MOGAs must be designed to appropriately ensure (1) the preservation of diversity and (2) the proper definition of a convergence criterion (Laumanns et al. 2002, Kumar and Pocket 2002). The first issue addresses the desire to identify as much of the Pareto front as possible (rather than getting stuck in a single section of it). Thus, a variety of substantially different Pareto points must be identified. Second, in most real-world problems, the true Pareto front is unknown, hampering to ascertain convergence. A final set of solutions might be non-dominated (and diverse) in relation to the GA population sample, but this does not necessarily imply its affiliation to the real, global Pareto front. Consequently, single-objective convergence criteria like "stop after a certain number of generations" or "stop after a certain portion of the population is non-dominated" are inappropriate. While increasing the run duration improves solution quality and raises probability of reaching the Pareto front, computational time might be wasted if the population has already converged.

Part II

MODELING AND SIMULATION OF PD PROCESSES

4. A New Process Model and Simulation

Previous models used to calculate PD process duration and cost rely on assumptions which do not capture the real complexity. As a consequence, a more thorough investigation of time-cost tradeoffs in PD processes requires a process model that addresses both, inter-activity (process architecture, overlapping policy, work policy) and intra-activity (crashing policy) effects. While many features of this model may have appeared in a previous study, no work has analyzed them all comprehensively as a system. Such a model is generally too complex to lend itself to closed-form analytical evaluation and must be rely on simulation techniques in order to approximate process cost and time. This chapter describes the proposed model that overcomes the aforementioned limitations and its analysis via discrete-time Monte Carlo simulation.

4.1 Process Architecture

Recalling definition 2.1, a process architecture describes the process activities and their dependency relationships (Browning and Eppinger 2002). Thereby, a process consists of n_A disjoint activities, which are recorded in a certain process sequence in vector $\boldsymbol{S}$. Due to the variety of relationship types between activities, the new PD process model must handle independent, unidirectional and bidirectional flows of information between activities. For this purpose, we use two numerical DSMs, each with real number values in the interval [0,1] to record these three types of information flow. A first DSM $M_{1,ij}$ records the probability of activity j causing rework for activity i. Since the intensity of rework for an activity can vary depending on its robustness – an effect known as rework impact (Browning and Eppinger 2002) – a further DSM $M_{2,ij}$ is necessary in order to record the fraction of activity i that must be reworked due to the output of activity j. For instance, $m_{2,ij} = 0.9$ means that in the event of rework for activity i, provoked by activity j, 90% of activity i must be repeated (Figure 4.1).

4.2 Activity Cost and Duration

As a next step, let us introduce the notation for an activity's cost and duration. In reality, both values are uncertain and cannot be assumed to be deterministic. As a triangular distribution for cost and duration is relatively easy to build with minimal data, we apply it for each activity yielding sampled values for cost, c_i, and duration, t_i, which are recorded in vectors $\boldsymbol{C}$ and $\boldsymbol{D}$ for all activities. The triangular distribution is a continuous probability distribution

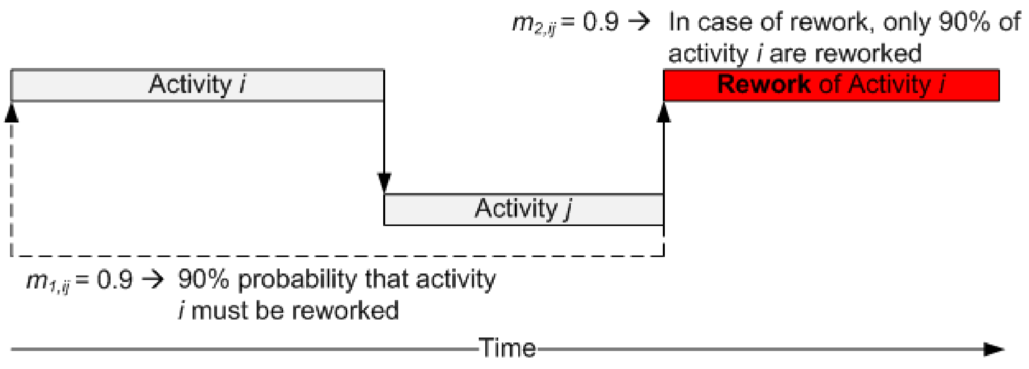
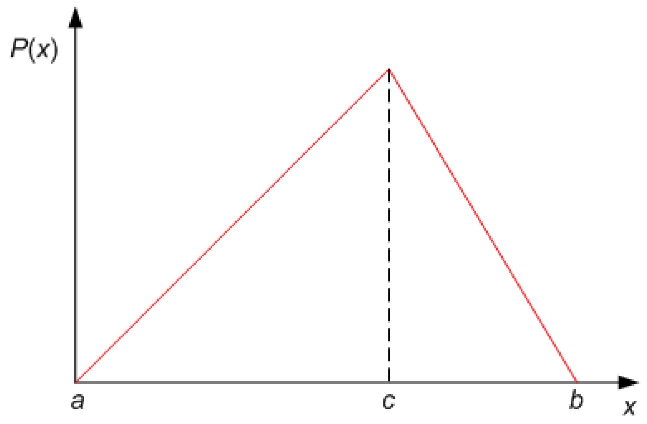

$m_{2,ij} = 0.9 \rightarrow$ In case of rework, only 90% of activity i are reworked
Activity i
Rework of Activity i
Activity j
$m_{1,ij} = 0.9 \rightarrow$ 90% probability that activity i must be reworked
Time

Figure 4.1: Demonstration of rework probability and rework impact.



Probability Density Function

$$P(x) = \begin{cases} \dfrac{2(x-a)}{(b-a)(c-a)} & \text{if } a \le x \le c \\ \dfrac{2(b-x)}{(b-a)(b-c)} & \text{if } c < x \le b \end{cases}$$

Cumulative Distribution Function

$$D(x) = \begin{cases} \dfrac{(x-a)^2}{(b-a)(c-a)} & \text{if } a \le x \le c \\ 1-\dfrac{(b-x)^2}{(b-a)(b-c)} & \text{if } c < x \le b \end{cases}$$

Figure 4.2: Illustration of the triangular PDF defined on the range $x \in [a,b]$ and related functions.

with a lower limit *a*, mode *c* and upper limit *b*. Figure 4.2 depicts the corresponding probability density function (PDF). In our case the following three data points are required for the PDF: minimal (optimistic) duration or cost, $\breve{t}_i$ and $\breve{c}_i$, most likely duration or cost, $\bar{t}_i$ and $\bar{c}_i$, and maximal (pessimistic) duration or cost, $\hat{t}_i$ and $\hat{c}_i$. Importantly, we sampled cost and duration values from correlated triangular PDFs using the Latin Hypercube Sampling (LHS) (Iman et al. 1981) to avoid unrealistic sampling pairs for cost and duration of the same activity. Basically, this strategy divides the range of values for c_i and t_i into equally probable intervals and accepts only sampled cost & duration values within the same interval. Thus, a duration value close to $\breve{t}_i$ entails a cost value close to $\breve{c}_i$ (and not close to $\hat{c}_i$), too.

However, the original values for duration (t_i) and cost (c_i) sampled from the respective PDFs may be changed in the k_i-th iteration of activity *i*, with $k_i \in [0,\infty)$, to $t_i(k_i)$ and $c_i(k_i)$ through crashing, learning curve effects, rework impact and overlapping. Thereby, $\tilde{t}_i(k_i)$ and $\tilde{c}_i(k_i)$ denote the cost and duration in iteration k_i considering rework impact and the effects of a learning curve which we assume to be constant in every iteration. These effects can arise if activity *i* is iterated (i.e. $k_i > 0$) because of a change in the output of an activity *j*. Thus, we obtain the following definitions:

$$\tilde{c}_i\left(k_i\right)=c_i\cdot\lambda_{ij}{}^{k_i} \tag{4.1}$$

$$\tilde{t}_i\left(k_i\right)=t_i\cdot\lambda_{ij}{}^{k_i} \tag{4.2}$$

with $\lambda_{ij}=l_i\cdot m_{2,ij}$ where $l_i\in\boldsymbol{L}$ indicates the learning curve value of activity *i* for both cost and duration, and $m_{2,ij}$ the rework impact. Consequently, we model learning as a function where an activity *i* takes l_i % of the original duration & cost in the second and subsequent executions.

Moreover, $\Delta c_{C_i(k_i)}$ and $\Delta t_{C_i(k_i)}$ record the change in cost and duration through crashing which we allow to be different in every single iteration. Contrary, $\Delta c_{O_i(k_i)}$ and $\Delta t_{O_i(k_i)}$ refer to the variable change in cost and lead time of activity *i* in iteration k_i as a result of overlapping with chronological predecessors. Additionally, an activity may be partially reworked in cyclic process environments influencing its cost and duration by $\Delta c_{R_i(k_i)}$ and $\Delta t_{R_i(k_i)}$. All these thoughts lead to the following two essential equations for cost and duration of an activity *i* in any of its iterations:

$$c_i\left(k_i\right)=\tilde{c}_i\left(k_i\right)+\Delta c_{O_i(k_i)}+\Delta c_{C_i(k_i)}+\Delta c_{R_i(k_i)} \tag{4.3}$$

$$t_i\left(k_i\right)=\tilde{t}_i\left(k_i\right)+\Delta t_{O_i(k_i)}+\Delta t_{C_i(k_i)}+\Delta t_{R_i(k_i)} \tag{4.4}$$

In the following sections we will describe the calculation of $\Delta c_{O_i(k_i)}$, $\Delta c_{R_i(k_i)}$, $\Delta c_{C_i(k_i)}$, $\Delta t_{O_i(k_i)}$, $\Delta t_{R_i(k_i)}$ and $\Delta t_{C_i(k_i)}$. Notably, an activity *i* features in every iteration a start time for execution, $T_{s_i(k_i)}$, and a finish time, $T_{f_i(k_i)}$. Thus, $t_i\left(k_i\right)=T_{f_i(k_i)}-T_{s_i(k_i)}$ holds. In any iteration, start time $T_{s_i(k_i)}$ and finish time $T_{f_i(k_i)}$ for any activity must be determined. The calculation of both dates is a matter of great importance and not trivial as many factors influence the computation. Finally, the overall cost and duration of the PD process are expressed by the parameters c_{tot} and t_{tot}, respectively.

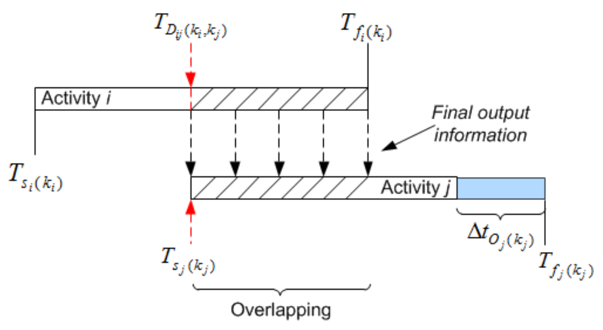
4.3 Activity Overlapping

Basically, the overlapping strategy schedules the time when an activity receives the input information from predecessors and when it delivers its output to successive activities thereby defining the duration of overlapping between an activity and any of its dependent activities (predecessors or successors).

Base case: overlapping based on finish to start relationships

In order to carefully establish all the overlapping parameters in our model, we initially assume simple finish to start relationships between dependent activities *i* and *j*. In this case, all output of *i* for *j* occurs at $T_{f_i(k_i)}$ and all input information for *j* is needed at $T_{s_j(k_j)}$. However, by assuming preliminary output information from an upstream activity *i*, the downstream activity *j* may be intentionally started earlier in time than $T_{f_i(k_i)}$ thereby overlapping with *i*. Although a reduction of the overall time span can be achieved in this way, overlapping may elongate the lead time of *j* by a penalty time $\Delta t_{O_j(k_j)}$ since it is started with imperfect predecessor information from *i* thus causing additional rework (Figure 4.3).

Thereby, the duration of overlapping between two dependent activities is mainly governed by two points in time (Figure 4.4): 1) the point in time when an upstream activity *i* delivers output information in the k_i-th iteration to a downstream activity *j* in the k_j-th



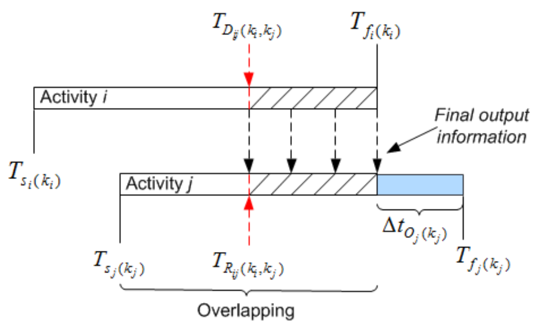


Figure 4.3: A simple overlapping of two activities with $T_{D_{ij}(k_i,k_j)} = T_{s_j(k_j)}$

Figure 4.4: A simple overlapping of two activities with $T_{D_{ij}(k_i,k_j)} = T_{R_{ij}(k_i,k_j)} \neq T_{s_j(k_j)}$

iteration, $T_{D_{ij}(k_i,k_j)}$ with $T_{D_{ij}(k_i,k_j)} \leq T_{f_i(k_i)}$, and 2) the point in time $T_{R_{ij}(k_i,k_j)}$ with $T_{R_{ij}(k_i,k_j)} \geq T_{s_j(k_j)}$ when a downstream activity *j* receives input in the k_j-th iteration from an upstream activity *i*.[15] Both dates are equal, i.e. $T_{D_{ij}(k_i,k_j)} = T_{R_{ij}(k_i,k_j)}$, but we still have to define these two points in time as their relationship to the start/finish time of their respective activity may differ. Thus, with respect to the base case, overlapping takes place if $T_{s_j(k_j)} < T_{D_{ij}(k_i,k_j)} < T_{f_j(k_j)}$ or $T_{f_i(k_i)} > T_{R_{ij}(k_i,k_j)} > T_{s_i(k_i)}$ holds (Figure 4.3 and Figure 4.4). Please note the extended rework duration in Figure 4.3 and Figure 4.4 as a consequence of an augmented overlapping duration.

Extending the base case by points in time for information available and needed

In the previous section, we assumed finish to start relationships between dependent activities. Though, it is more general (and more realistic) to assume that the complete output information of an upstream activity *i* for a downstream activity *j* is available at a point in time $T_{A_{ij}(k_i,k_j)}$ with $T_{A_{ij}(k_i,k_j)} \leq T_{f_i(k_i)}$ and vice versa that an activity *i* needs input information at a point in time $T_{N_{ij}(k_i,k_j)}$ with $T_{N_{ij}(k_i,k_j)} \geq T_{s_i(k_i)}$. Since $T_{D_{ij}(k_i,k_j)} \leq T_{A_{ij}(k_i,k_j)} \leq T_{f_i(k_i)}$ holds, $T_{A_{ij}(k_i,k_j)}$ constitutes an upper bound for $T_{D_{ij}(k_i,k_j)}$ while $T_{N_{ij}(k_i,k_j)}$ constitutes a lower bound for $T_{R_{ij}(k_i,k_j)}$ due to $T_{R_{ij}(k_i,k_j)} \geq T_{N_{ij}(k_i,k_j)} \geq T_{s_j(k_j)}$ (Figure 4.5). In order to determine $T_{A_{ij}(k_i,k_j)}$, we simply need to add

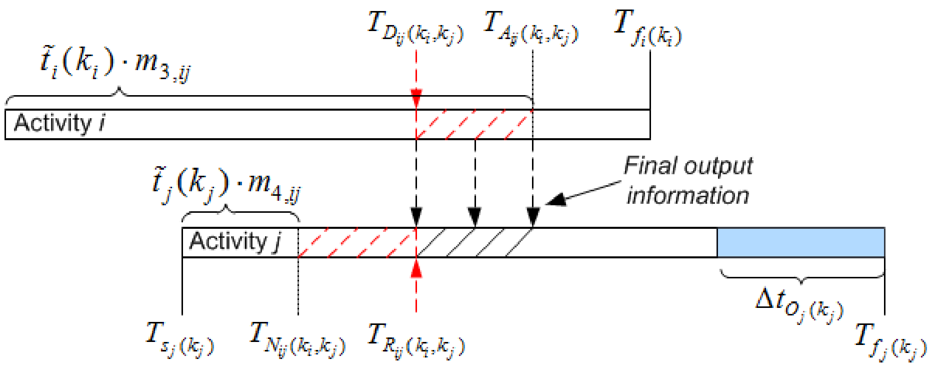


Figure 4.5: Overlapping between two activities including $T_{A_{ij}(k_i,k_j)}$ and $T_{N_{ij}(k_i,k_j)}$

[15] For cyclic processes, overlapping may also occur between a downstream activity feeding an upstream one.

the percent of activity *i* required before its complete output is available, recorded in $M_{3,ij}$, to its start date: $T_{A_{ij}\left(k_i,k_j\right)} = T_{s_i\left(k_i\right)} + \tilde{t}_i\left(k_i\right) \cdot m_{3,ij}$. Similar, $T_{N_{ij}\left(k_i,k_j\right)}$ is calculated through the summation of the percent of activity *j* which can occur before it requires complete input from activity *i*, recorded in $M_{4,ij}$, with the start time of activity *j*: $T_{N_{ij}\left(k_i,k_j\right)} = T_{s_i\left(k_i\right)} + \tilde{t}_i\left(k_i\right) \cdot m_{4,ij}$.

Bounds for preliminary information available and needed

Since we like to exactly specify the theoretical time period in which $T_{D_{ij}\left(k_i,k_j\right)}$ and $T_{R_{ij}\left(k_i,k_j\right)}$ can occur, we need to furthermore define a lower bound for $T_{D_{ij}\left(k_i,k_j\right)}$ as well as an upper bound for $T_{R_{ij}\left(k_i,k_j\right)}$. Thereby, $T_{D_{ij}\left(k_i,k_j\right)}$ is bounded below by a time point constituting the percent of activity *i*'s duration required before its preliminary output is available for a downstream activity *j*. In contrast, $T_{R_{ij}\left(k_i,k_j\right)}$ is bounded above by a point in time representing the percent of activity *j* that can occur before it requires preliminary input from its upstream activity *i*.

The former percent value is recorded in M_5 and consequently: $T_{s_i\left(k_i\right)} + \tilde{t}_i\left(k_i\right) \cdot m_{5,ij} \leq T_{D_{ij}\left(k_i,k_j\right)}$. Accordingly, rework penalties due to faulty predecessor information are assigned to activity *j* if $T_{D_{ij}\left(k_i,k_j\right)} \in \left[T_{s_i\left(k_i\right)} + \tilde{t}_i\left(k_i\right) \cdot m_{5,ij},\ T_{A_{ij}\left(k_i,k_j\right)}\right]$. On the other hand, the percent values for the upper bound of $T_{R_{ij}\left(k_i,k_j\right)}$ can be found in M_6, so that $T_{R_{ij}\left(k_j\right)} \leq T_{s_j\left(k_j\right)} + \tilde{t}_j\left(k_j\right) \cdot m_{6,ij}$ holds and activity *j* is penalized with rework if $T_{R_{ij}\left(k_i,k_j\right)} \in \left[T_{N_{ij}\left(k_i,k_j\right)},\ T_{s_j\left(k_j\right)} + \tilde{t}_j\left(k_j\right) \cdot m_{6,ij}\right]$. Figure 4.6 visualizes the newly introduced bounds and further overlapping related parameters presented in the next paragraph.

Final calculation for overlapping duration

With these bounds established, we can now exactly compute $T_{D_{ij}\left(k_i,k_j\right)}$ and $T_{R_{ij}\left(k_i,k_j\right)}$ in any

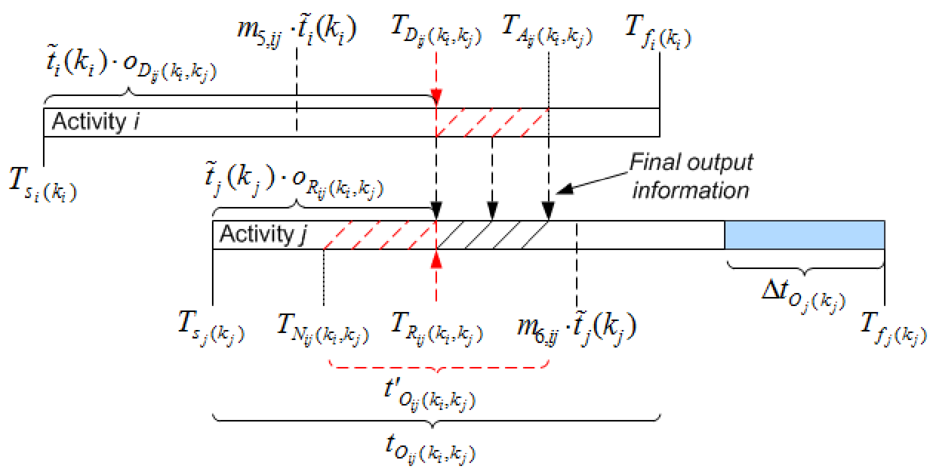


Figure 4.6: Overlapping between two activities displaying all relevant overlapping parameters.

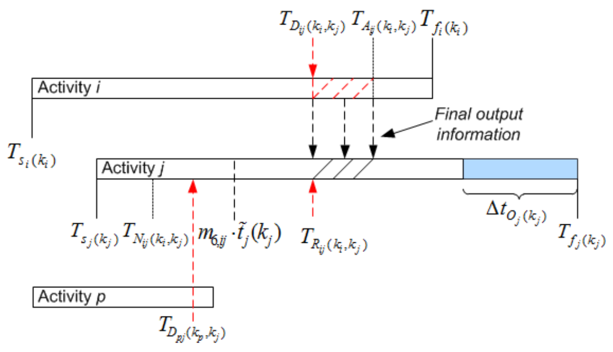
iteration using the parameters $o_{D_{ij}(k_i,k_j)} \in [m_{5,ij}, m_{3,ij}]$ and $o_{R_{ij}(k_i,k_j)} \in [m_{4,ij}, m_{6,ij}]$ representing the percent of an activity which must be processed until it delivers/receives information. The following equations hold:

$$T_{D_{ij}(k_i,k_j)} = T_{s_i(k_i)} + \tilde{t}_i(k_i) \cdot o_{D_{ij}(k_i,k_j)} \quad \text{, if activity } j \text{ is not active} \tag{4.5}$$

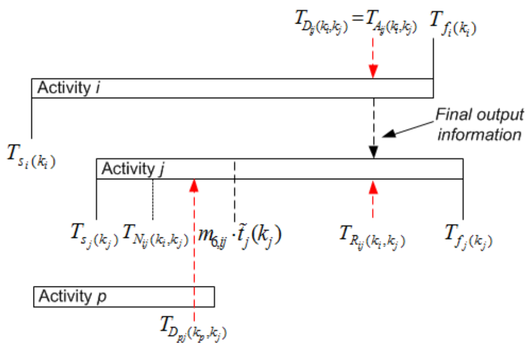
$$T_{D_{ij}(k_i,k_j)} = T_{A_{ij}(k_i,k_j)} \quad \text{, if activity } j \text{ is active} \tag{4.6}$$

$$T_{R_{ij}(k_j)} = T_{s_j(k_j)} + \tilde{t}_j(k_j) \cdot o_{R_{ij}(k_i,k_j)} \tag{4.7}$$

Beside the obvious equations 4.5 and 4.7, equation 4.6 accounts for a special case: an activity *i* is supposed to deliver its preliminary output information to an activity *j* which is *active*, i.e. activity *j* has already started execution before $T_{D_{ij}(k_i,k_j)}$, due to the output of another activity *p* (Figure 4.7a). In this event, it is reasonable (and economical) to wait with the transfer of information from *i* to *j* until the final output of activity *i* is available, i.e. $T_{D_{ij}(k_i,k_j)} = T_{A_{ij}(k_i,k_j)}$ (Figure 4.7b), instead of feeding *j* with preliminary output - although possible in theory. Otherwise, the poor quality of *i*'s preliminary information would simply provoke additional rework of *j* without any temporal benefits (e.g. a reduction of overall time for *i* and *j* combined). Thus, we do not refer to the described case as an overlapping situation.



(a) Activity *p* causes the execution of *j* before activity *i* delivers output to *j*.



(b) Delay of preliminary output information of *i* in order to avoid unnecessary rework for *j*.

Figure 4.7: Special overlapping case: downstream activity *j* is active.

Next, we present the computation of overlapping durations $t_{O_{ij}(k_i,k_j)}$ and $t'_{O_{ij}(k_i,k_j)}$ for the k_i-th iteration of activity *i* and the k_j-th iteration of activity *j*. While the entire overlapping duration between activity *i* and *j* is described by $t_{O_{ij}(k_i,k_j)}$, $t'_{O_{ij}(k_i,k_j)}$ constitutes the overlapping duration which causes rework for activity *j* due to an imperfect information flow between *i* and *j*:

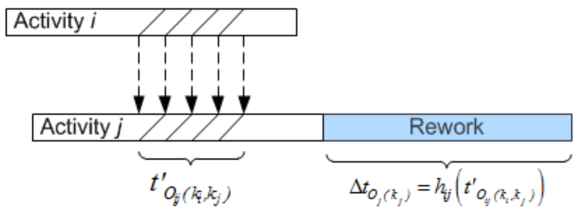
$$t_{O_{ij}(k_i,k_j)} = \min\left\{T_{f_i(k_i)}, T_{f_j(k_j)}\right\} - \max\left\{T_{s_i(k_i)}, T_{s_j(k_j)}\right\} \tag{4.8}$$

$$t'_{O_{ij}(k_i,k_j)} = \min\left\{T_{A_{ij}(k_i,k_j)}, T_{f_j(k_j)}\right\} - \max\left\{T_{s_i(k_i)}, T_{N_{ij}(k_i,k_j)}\right\} \quad \text{, if activity } j \text{ is not active} \tag{4.9}$$

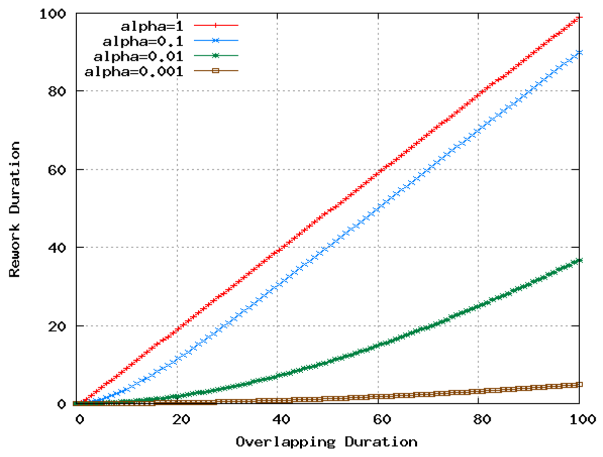
$$t'_{O_{ij}(k_i,k_j)} = 0 \quad \text{, if activity } j \text{ is active} \tag{4.10}$$

Rework duration caused by overlapping

In order to determine the amount of rework generated through overlapping we still have to define an overlapping function $h_{ij}\left(t'_{O_{ij}(k_i,k_j)}\right)$. This function calculates the time span of rework between an activity *j* and one of its chronologically precedent activities *i* as a function of the overlapping duration $t'_{O_{ij}(k_i,k_j)}$ (Figure 4.8a). The set of all predecessors for an activity *j* in the k_j-th iteration is denoted as $\boldsymbol{P}_j$. Similar to Roemer et al. (2004) we define a first overlapping function as follows:


Activity *i*
Activity *j*
Rework
$t'_{O_{ij}(k_i,k_j)}$
$\Delta t_{O_j(k_j)} = h_{ij}\left(t'_{O_{ij}(k_i,k_j)}\right)$

(a) Effect of the overlapping function.



(b) Plot of the overlapping function.

Figure 4.8: Illustration of rework through overlapping.

$$h'_{ij}\left(t'_{O_{ij}(k_i,k_j)}\right) = t'_{O_{ij}(k_i,k_j)} - \frac{1-e^{-\alpha_{ij}\cdot t'_{O_{ij}(k_i,k_j)}}}{\alpha_{ij}} \tag{4.11}$$

with $0 < \alpha_{ij} < \infty$. The corresponding function is depicted in Figure 4.8b for different α_{ij} values. Moreover, we define a second, linear, overlapping function with $0 < \alpha_{ij} < \infty$:

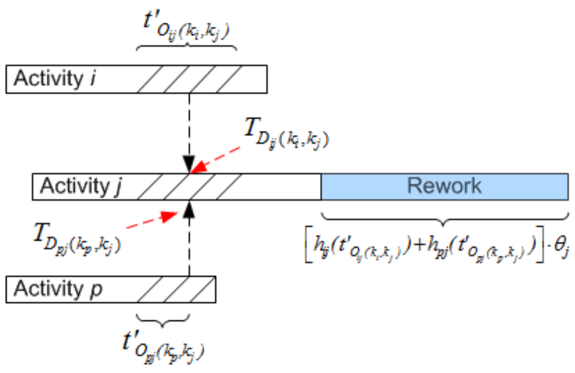
$$h''_{ij}\left(t'_{O_{ij}(k_i,k_j)}\right) = \alpha_{ij} \cdot t'_{O_{ij}(k_i,k_j)} \tag{4.12}$$

The linear overlapping function is probably more intuitive for practitioners as it is easier to immediately calculate the consequences of overlapping (i.e. the rework duration/cost).

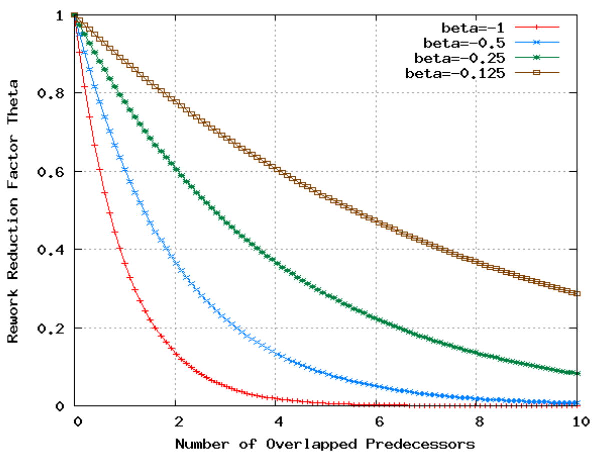
Simultaneous overlapping events

It is possible that two or more chronological predecessors of an activity *j* overlap with *j* and deliver their information to *j* at the same time (Figure 4.9a). In this event, we do not simply cumulate the individual rework durations between *j* and any $i \in \boldsymbol{P}_j$. Rather, we reduce the sum of every pair-wise rework between an activity *j* and all its overlapping predecessors by a factor θ_j depending on the cardinality of $\boldsymbol{P}_j$, where θ_j is calculated as follows:

$$\theta_j = e^{\beta\cdot|\boldsymbol{P}_j|} \tag{4.13}$$



(a) Overlapping of three activities.



(b) Plot of the rework reduction factor.

Figure 4.9: Overlapping of multiple activities.

with $\beta \in (-\infty, 0]$ (Figure 4.9b). As we assume the maximal pair-wise rework between j and any $i \in \boldsymbol{P}_j$ to be the minimal amount of cumulative rework, we must furthermore define a predecessor $\hat{i} \in \boldsymbol{P}_j$ with maximal rework time $\hat{h}_{ij}\left(t'_{O_{ij}(k_i,k_j)}\right) = \max\left\{h_{ij}\left(t'_{O_{ij}(k_i,k_j)}\right) \mid \forall i \in \boldsymbol{P}_j\right\}$. Hence, overall cumulative rework $\Delta t_{O_j(k_j)}$ for a downstream activity j due to overlapping constitutes:

$$\Delta t_{O_j(k_j)} = \hat{h}_{ij}\left(t'_{O_{ij}(k_i,k_j)}\right) + \theta_i \cdot \left(\sum_{\forall p \in \boldsymbol{P}_j, p \neq \hat{i}} h_{ij}\left(t'_{O_{ij}(k_i,k_j)}\right)\right) \tag{4.14}$$

Rework cost caused by overlapping

Finally, we assume a proportional relationship between the rise in cost of any activity i due to rework caused by overlapping with its predecessors in iteration k_i, denoted as $\Delta c_{O_i(k_i)}$, and the duration of rework $\Delta t_{O_i(k_i)}$: $\Delta c_{O_i(k_i)} \sim \Delta t_{O_i(k_i)}$. Accordingly, $\Delta c_{O_i(k_i)}$ is a percentage of activity i's cost considering learning and rework effects, proportional to the ratio of overlapping rework duration and duration prior to overlapping:

$$\Delta c_{O_i(k_i)} = \tilde{c}_i(k_i) \cdot \frac{\Delta t_{O_i(k_i)}}{\tilde{t}_i(k_i)} \tag{4.15}$$

4.4 Activity Crashing

Time reduction through crashing

To reduce the time of an activity *i*, it can be crashed in the k_i-th iteration by a duration of $\Delta t_{C_i(k_i)}$ to $\left(\tilde{t}_i(k_i)+\Delta t_{O_i(k_i)}\right)-\left(\tilde{t}_i(k_i)+\Delta t_{O_i(k_i)}\right)\cdot\varpi_i(k_i)$. Thereby, we define $\varpi_i(k_i)$ as follows:

$$\varpi_i(k_i)=r_i(k_i)\cdot\kappa_i \tag{4.16}$$

where $r_i(k_i)\in[0,\hat{r}_i]$ corresponds to the actual crashing intensity whereas $\hat{r}_i\in[0,1)$ represents the maximal crashing intensity recorded in vector ***R***, and κ_i constitutes a crashing factor which we refer to as *Brooks factor*. The Brooks factor κ_i is either negative or 1.0, i.e. $(\kappa_i<0)\vee(\kappa_i=1)$, and allows us to model situations where activities resist crashing and can be, at worst, elongated despite the assignment of large resources (Brooks 1995). Obviously, $\tilde{t}_i(k_i)+\Delta t_{O_i(k_i)}$ is either decreased through crashing if $0<\varpi_i(k_i)<1$ holds, or remains unchanged if $\varpi_i(k_i)=0$, or is increased if $\varpi_i(k_i)<0$. Figure 4.10 illustrates the effects of $r_i(k_i)$ and κ_i on an activities' duration. Hence, the change in duration of activity *i* due to crashing in iteration k_i can be expressed as:

$$\Delta t_{C_i(k_i)}=-\left(\tilde{t}_i(k_i)+\Delta t_{O_i(k_i)}\right)\cdot\varpi_i(k_i) \tag{4.17}$$

Unlike past literature (Roemer and Ahmadi 2004, Roemer et al. 2000), which mainly investigated the crashing of entire process stages, we do not allow a varying crashing intensity over time while the activity is executed. We feel this is a reasonable assumption since we decompose the entire PD process in much smaller work units and the duration of a single activity constitutes only a fraction of the lead time for an entire process stage (e.g., concept

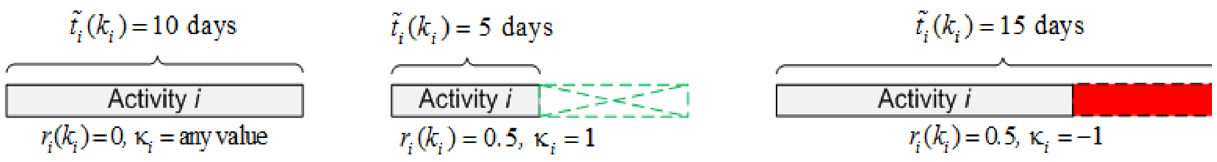

$\tilde{t}_i(k_i)=10$ days
Activity *i*
$r_i(k_i)=0,\ \kappa_i=$ any value
$\tilde{t}_i(k_i)=5$ days
Activity *i*
$r_i(k_i)=0.5,\ \kappa_i=1$
$\tilde{t}_i(k_i)=15$ days
Activity *i*
$r_i(k_i)=0.5,\ \kappa_i=-1$

Figure 4.10: The effects of crashing.

development). Hence, it is realistic to assume that a single activity is merely crashed with a constant intensity. Though, we allow a varying crashing intensity in different iterations of an activity.

General cost increase through crashing

As mentioned in the introductory section, the immediate side-effect of crashing an activity is its augmented cost. Thereby, an activity's cost rise due to crashing in the k_i-th iteration, $\Delta c_{C_i(k_i)}$, depends on its corresponding crashing intensity $r_i(k_i)$ and a crashing function, $R_i(r_i(k_i))$, which we allow to be continuous or discrete. Generally, we note:

$$\Delta c_{C_i(k_i)} = \left(\tilde{c}_i(k_i) + \Delta c_{O_i(k_i)}\right) \cdot R_i\left(r_i(k_i)\right) \tag{4.18}$$

Thus, the change in cost as a consequence of crashing is a percentage of the activity's cost in iteration k_i. In the continuous case we set:

$$R_i\left(r_i(k_i)\right) = \left(\frac{1}{\left(1-r_i(k_i)\right)^{\alpha}} - 1\right) \Big/ 100, \ \alpha \in [0, \infty) \tag{4.19}$$

Figure 4.11a plots the convex function $R_i\left(r_i(k_i)\right)$ for different values of α. However, many activities have only particular ways in which they can be done differently - e.g., by using technology B instead of technology A. This calls for a step function instead. Given a sequence

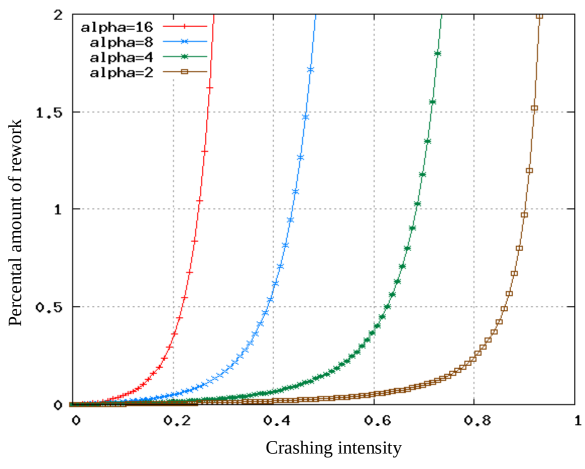


Figure 4.11: Illustration of the continuous crashing function for different values of α.

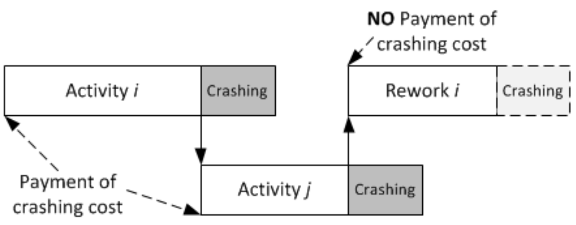


Figure 4.12: Illustration of crashing resources being paid only once.

of m coefficients $\{\alpha_0, \alpha_1, \ldots, \alpha_m\} \subset \mathbb{R}$ and a sequence of interval margins $\{x_1, x_2, \ldots x_{m-1}\} \subset [0,1)$ we can define a sequence of intervals: $A_0 = [0, x_1)$, $A_s = [x_s, x_{s+1}), \forall s = 1,2\ldots,m-2$ and $A_m = [x_{m-1}, 1)$. With these parameters we define a second crashing function:

$$R_i'\left(r_i\left(k_i\right)\right) = \left(\sum_{s=0}^{m}\left(\frac{1}{\left(1-r_i\left(k_i\right)\right)^{\alpha_s}} - 1\right) \cdot 1_{A_s}\left(r_i\left(k_i\right)\right)\right) \Big/ 100 \tag{4.20}$$

with 1_A as the indicator function of A, i.e.:

$$1_A\left(r_i\left(k_i\right)\right) = \begin{cases} 1, \text{ if } r_i\left(k_i\right) \in A_s \\ 0, \text{otherwise} \end{cases} \tag{4.21}$$

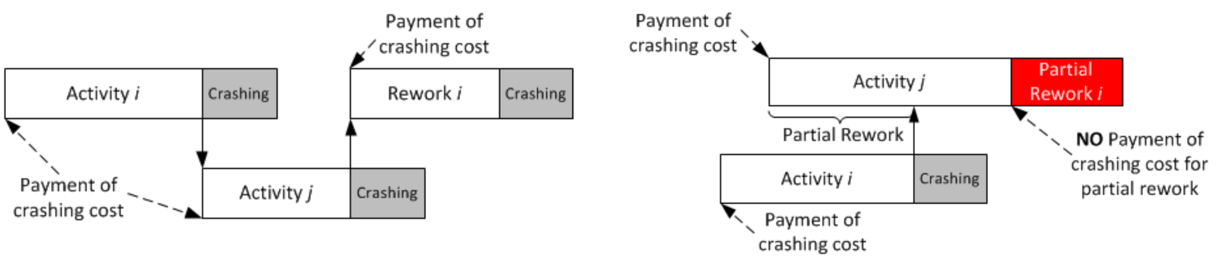
Crashing cost for cyclic processes

In practice, the effects of crashing are achieved through an assignment of additional resources which may be paid in different ways. While irrelevant for acyclic processes, the payment mode of extra resources affects $\Delta c_{C_i(k_i)}$ of cyclic processes. We distinguish the following modes:

1. *One-time resource cost* which must be paid only once in the initial iteration since they are existent for the remaining process duration (e.g. machinery). In this case,

$$\Delta c_{C_i(k_i)} = \begin{cases} \left(\tilde{c}_i\left(k_i\right) + \Delta c_{O_i(k_i)}\right) \cdot R_i\left(r_i\left(k_i\right)\right), & \text{if } k_i = 0 \\ 0, & \text{if } k_i > 0 \end{cases} \tag{4.22}$$

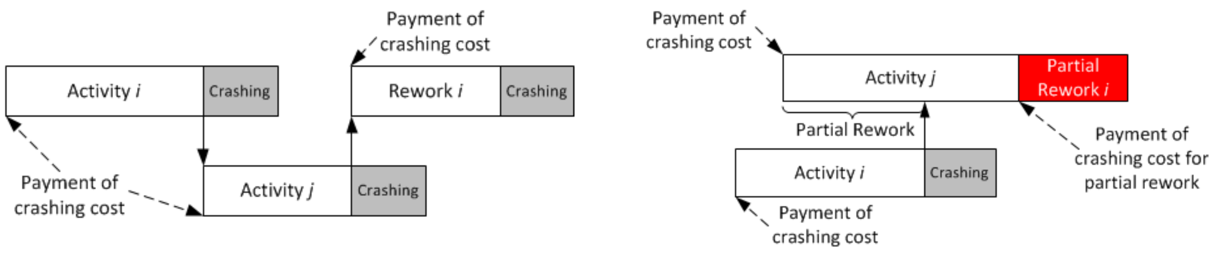
holds (see Figure 4.12).



(a) Crashing cost must be paid in every iteration including rework.

(b) Crashing cost for partial rework is not paid.

Figure 4.13: Illustration of single payment crashing.



(a) Crashing cost must be paid in any iteration including rework.

(b) Crashing cost for partial rework must be paid.

Figure 4.14: Illustration of multiple payment crashing.

2. *Single payment resource cost* which are paid only once at the beginning of every iteration (e.g. set-up cost for machinery, rental of facilities) including reworked activities (Figure 4.13a). Importantly in this case, partial rework (see next subsection) of an activity in any iteration will not affect its crashing cost since $\Delta c_{C_i(k_i)}$ is computed according to equation 4.18 only once for any new iteration (Figure 4.13b).
3. Finally, we define *multi payment cost* for resources which must be paid continuously during activity execution in any iteration (e.g. staff). Consequently, crashing cost must be considered for an activity each time it is started (Figure 4.14a) and for partial rework cost (Figure 4.14b; see next subsection). Again, $\Delta c_{C_i(k_i)}$ is determined, subject to equation 4.18, when it is started in any new iteration.

4.5 Partial Rework of Activities

Base Case: Singular partial rework provoked by one activity

Sometimes, a downstream activity *j* may deliver its output to an upstream activity *i* after the latest point in time when activity *i* needed it, i.e. $T_{D_{ji}(k_j,k_i)} > m_{6,ji} \cdot \tilde{t}_i(k_i)$. Such a situation typically emerges if $T_{f_j(k_j)} < T_{f_i(k_i)}$ and $m_{1,ij}$ with $j>i$ holds, which is only possible for cyclic processes. Iterative overlapping (see section 3.1 and definition 3.1) would be a practical example for this case. In this event, the duration $T_{D_{ij}(k_i,k_j)} - T_{N_{ij}(k_i,k_j)}$ must be reworked as it

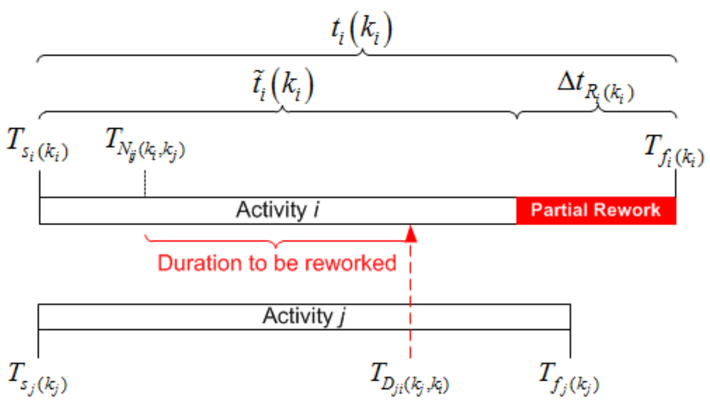


Figure 4.15: Illustration of singular partial rework.

constitutes the time span between information delivery and the lower bound for information needed. We refer to this reworked duration as *partial rework*.

As a consequence of partial rework, cost and duration of an activity *i* (i.e. $c_i(k_i)$ and $t_i(k_i)$) are modified by $\Delta c_{R_i(k_i)}$ and $\Delta t_{R_i(k_i)}$. If activity *i* is partially reworked for the first time, caused by an activity *j*, the temporal change $\Delta t_{R_i(k_i),1}$ (the “1” denotes the calculation of partial rework of *i* for the first time) is calculated as follows:

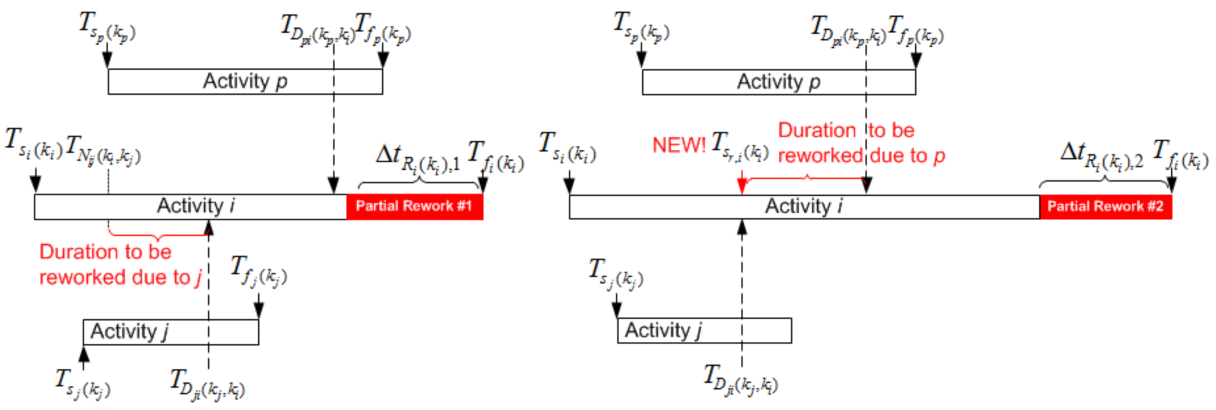
$$\Delta t_{R_i(k_i),1} = \left(T_{D_{ji}(k_j,k_i)} - T_{N_{ij}(k_i,k_j)}\right) \cdot \lambda_{ij} \tag{4.23}$$

Figure 4.15 illustrates the above calculation. It is important to mention that $T_{f_i(k_i)}$ changes due to the added duration. Contrary, the calculation of $\Delta c_{R_i(k_i),1}$ depends on how crashing resources are paid. If crashing cost are paid via single payment (case 1 and 2 in the previous section) we note:

$$\Delta c_{R_i(k_i),1} = \frac{\left(T_{D_{ji}(k_j,k_i)} - T_{N_{ij}(k_i,k_j)}\right) \cdot \lambda_{ij}}{T_{f_i(k_i)} - T_{s_i(k_i)}} \cdot \left(\tilde{c}_i(k_i) + \Delta c_{O_i(k_i)}\right) \tag{4.24}$$

Otherwise, if crashing cost must be paid continuously, the following equation holds:

$$\Delta c_{R_i(k_i),1} = \frac{\left(T_{D_{ji}(k_j,k_i)} - T_{N_{ij}(k_i,k_j)}\right) \cdot \lambda_{ij}}{T_{f_i(k_i)} - T_{s_i(k_i)}} \cdot \left(\tilde{c}_i(k_i) + \Delta c_{O_i(k_i)} + \Delta c_{C_i(k_i)}\right) \tag{4.25}$$



$T_{s_p(k_p)}$
$T_{D_{pi}(k_p,k_i)}$ $T_{f_p(k_p)}$
Activity p
$T_{s_i(k_i)}$ $T_{N_{ij}(k_i,k_j)}$
$\Delta t_{R_i(k_i),1}$ $T_{f_i(k_i)}$
Activity i
Partial Rework #1
Duration to be reworked due to j
$T_{f_j(k_j)}$
Activity j
$T_{s_j(k_j)}$ $T_{D_{ji}(k_j,k_i)}$
NEW! $T_{s_{r,i}(k_i)}$
Duration to be reworked due to p
$\Delta t_{R_i(k_i),2}$
Partial Rework #2

(a) First partial rework due to activity *j*. (b) Second partial rework due to activity *p*.

Figure 4.16: Illustration of multiple partial rework.

Multiple partial rework

Generally, an activity *i* might be partially reworked $n_{r,i}$ times during its execution until it is completely reworked again, i.e. until k_i changes. Hence, we have to accumulate cost and duration associated with any partial rework in order to finally derive $c_i(k_i)$ and $t_i(k_i)$:

$$\Delta t_{R_i(k_i)} = \sum_{l=1}^{n_{r,i}} \Delta t_{R_i(k_i),l} \tag{4.26}$$

$$\Delta c_{R_i(k_i)} = \sum_{l=1}^{n_{r,i}} \Delta c_{R_i(k_i),l} \tag{4.27}$$

The calculation for case $n_{r,i}$=1 was pointed out in the previous paragraph. Though, if $n_{r,i}$>1 we have to change the temporal reference system in order to apply equations 4.23-4.25. Firstly, after an activity was partially reworked, its finish time must be appropriately updated. Moreover, we introduce a point in time for the information input which provoked the latest partial rework: $T_{s_{r,i}(k_i)}$. Hence, $T_{s_{r,i}(k_i)}$ must be updated every time when activity *i* is partially reworked. Figure 4.16 visualizes a scenario for two partial rework events with all important model parameters. Formally, we define:

$$\Delta t_{R_i(k_i),l} = \begin{cases} \left(T_{D_{ji}(k_j,k_i)} - T_{N_{ij}(k_i,k_j)}\right)\cdot \lambda_{ij} & \text{, if } l=1 \\ \left(T_{D_{ji}(k_j,k_i)} - T_{s_{r,i}(k_i)}\right)\cdot \lambda_{ij} & \text{, if } l>1 \end{cases} \tag{4.28}$$

$$\Delta c_{R_i(k_i),l} = \begin{cases} \dfrac{\left(T_{D_{ji}(k_j,k_i)} - T_{N_{ij}(k_i,k_j)}\right)\cdot\lambda_{ij}}{T_{f_i(k_i)} - T_{s_i(k_i)}} \cdot \left(\tilde{c}_i\left(k_i\right) + \Delta c_{O_i(k_i)} + \Delta c_{C_i(k_i)}\right) & \text{, if } l = 1 \\ \dfrac{\left(T_{D_{ji}(k_j,k_i)} - T_{s_{r,i}(k_i)}\right)\cdot\lambda_{ij}}{T_{f_i(k_i)} - T_{s_{r,i}(k_i)}} \cdot \left(\tilde{c}_i\left(k_i\right) + \Delta c_{O_i(k_i)} + \Delta c_{C_i(k_i)} + \sum_{z=1}^{l-1} \Delta c_{R_i(k_i),z}\right) & \text{, if } l > 1 \end{cases} \tag{4.29}$$

Notably, we considered the continuous crashing case (equation 4.25) regarding equation 4.29.

Simultaneous partial rework events

As opposed to overlapping (see section 4.3), the partial rework of an activity *i* caused through multiple activities at the same time does not require an advanced coordination effort between the individual activities. Thus, there is no need for a similar calculation as in case of simultaneous overlapping events using a rework reduction factor. Instead, we decided to separately calculate rework cost and durations if two or more activities deliver their output to an activity *i* at an identical point in time and cause partial rework of *i*. Out of these values we simply select the maximal values with respect to rework cost and duration and assign them to activity *i*.

4.6 Work Policies

In addition to the work policy proposed by Browning and Eppinger (2002), denoted as P_1, we defined four new work policies P_2-P_5 through the assignment of rules which are supposed to affect the process flow (and thus t_{tot} as well as c_{tot}) in different ways. An overview of the respective rules in combination with a visual example as well as their assignment to P_1-P_5 is depicted at Table 4.1. In practice, the entire set of rules of the defined work policies must not necessarily be valid for every activity within the process. For instance, some activities might be allowed to work simultaneously if they are not necessarily adjacent while others are not. Though, to avoid further complications of our process model, we resigned on such a "mix" of policy rules on an activity level. While most of the rules should be self-explanatory, we like to

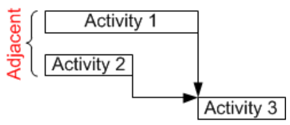
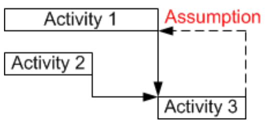
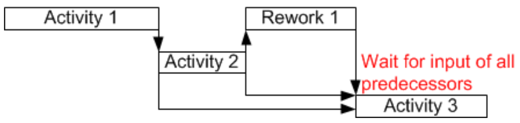
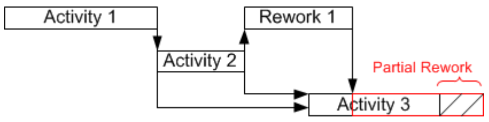
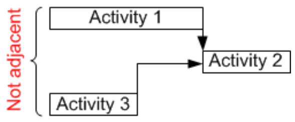
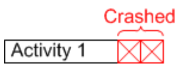
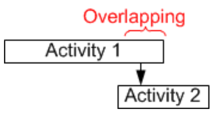
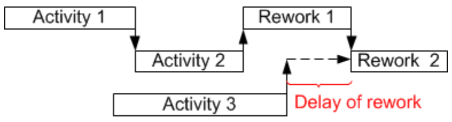
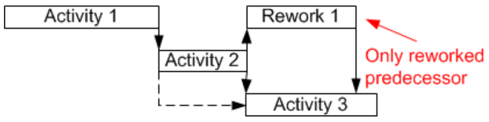
Rule	Policy 1	Policy 2	Policy 3	Policy 4	Policy 5
Adjacent activities may be executed concurrently: 	✓	✓	✓	✓	✓
Activities can make assumptions about its input from downstream activities: 	✓	✓	✓	✓	✓
All predecessor information must be available before an activity can start:  Activity 1 Rework 1 Activity 2 Wait for input of all predecessors Activity 3	✓	✓	✓	✓	✗
Partial rework is immediately accomplished: 	✓	✓	✓	✓	✓
Non-Adjacent activities are allowed to be executed concurrently: 	✗	✓	✓	✓	✓
An activity may be crashed: 	✗	✗	✓	✓	✓
Sequential activities may be overlapped: 	✗	✗	✓	✓	✓
Delay of an activity to be reworked according to some strategy:  Activity 1 Rework 1 Activity 2 Rework 2 Activity 3 Delay of rework	✗	✗	✗	✓	✗
Activities can start if all non-reworked predecessor information is available:  Activity 1 Rework 1 Activity 2 Only reworked predecessor Activity 3	✗	✗	✗	✗	✓

Table 4.1: Overview of rules assigned to the five work policies P_1-P_5.

dwell on the final two rules of Table 4.1.

Regarding the first case, we recognize that it might be sometimes beneficial to intentionally delay the rework of an activity *i* if one or more predecessors of *i* are still processed (look at the visual example in the corresponding cell of Table 4.1). Because these predecessors may deliver new input information to *i*, subject to a certain probability, they could all cause partial (or entire) rework of *i* which ultimately increments overall cost. Such a situation could be prevented through a delay of the start time of *i* according to some strategy. For instance, if *i* was supposed to start at $T_{s_i(k_i)}$ we decided to delay its beginning to the maximal finish time of its predecessors, $\hat{T}_{s_i(k_i)}$, which are still executed at $T_{s_i(k_i)}$: $\hat{T}_{s_i(k_i)} = \max\left\{T_{f_j(k_j)} \mid j \in \boldsymbol{P}_i : T_{f_j(k_j)} > T_{s_i(k_i)}\right\}$. Though, as downside of this strategy, time could be wasted if *i* is eventually not fed with new input information by its predecessors.

Considering the lowermost rule in Table 4.1, we intended to allow the start of an activity if all predecessors have been completed at least once. Therefore, an activity does not have to wait for the final output of reworked predecessors but may work in parallel of it. This rule is expected to result in a decrease of process duration but also in an increase of rework and thus cost.

Last but not least let us elaborate on the impact of activity sequence in dependence of work policy rules. Clearly, the different rules in Table 4.1 determine when an activity is allowed to start, in particular if predecessor information is required or not (compare P_1 and P_5 as extreme examples). Simultaneously, the activity sequence governs the definition of predecessors and successors. Consequently, we expect an interaction between these two managerial levers. In fact, fewer restrictions with respect to predecessor information needed are likely to increase the number of redundant activity sequences. Activities may be exchanged in the activity sequence much more often without affecting their actual start times if work policy rules allow a start independent of predecessor outputs. Therefore, we suspect less spread regarding process cost and duration due to changes in the activity sequences for minor restrictive work policies like P_5 and vice versa higher time-cost variances for highly restrictive work policies in terms of input information needed such as P_1.

4.7 Monte-Carlo Simulation for Process Duration and Cost

To investigate time-cost tradeoffs, we must compare varied instances of this model with respect to c_{tot} and t_{tot}. To gain realistic values for both measures, we simulate each instance with a discrete event, Monte Carlo simulation - a slight modification of the approach proposed by Browning and Eppinger (2002). Our Monte Carlo simulation employs the Latin Hypercube sampling (LHS) method (Iman et al. 1981) as the conditional Monte Carlo Sampling techniques (Adlakha 1986) do not help reduce computational effort in an iterative project since activities iterate probabilistically. Basically, the LHS method produces a paired c_{tot} and t_{tot} for each run which can then be used to form individual or joint probability density functions. The simulation can be set to terminate when either a certain number of runs has been executed or a stable state has been reached in the simulation results (i.e., if the mean and variance in the PDFs of output values falls below a certain threshold).

Calculation of the next event T_{E}

During simulation, we distinguish between a set of *active* activities, $\boldsymbol{H}$, and a set of *not active/inactive* activities $\boldsymbol{I}$. If an activity i is active, its finish time $T_{f_i(k_i)}$ is greater than the current simulation time while it is not active if its finish time is smaller than the current simulation time. In order to simulate time and cost for processes according to our model, we repeatedly have to calculate the point in time for the next event, T_{E}, until the simulation converges. Essentially, T_{E} is either 1) the minimal finish time of all not active activities $\breve{T}_I = \min\left\{T_{f_i(k_i)} \mid \forall i \in \boldsymbol{I}\right\}$ or 2) the earliest point in time when any active activity is allowed to

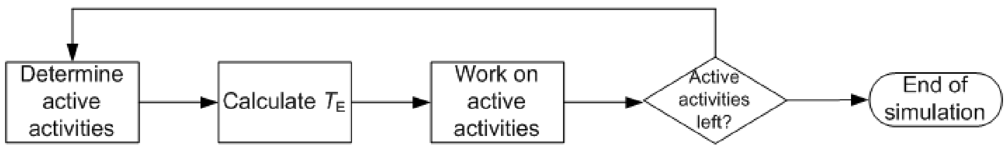


Figure 4.17: Simulation event graph.

deliver output to any other activity $\breve{T}_D = \min\left\{T_{D_{ij}(k_i,k_j)} \mid \forall i \in \boldsymbol{H}\right\}$. More formally:

$$T_E = \min\left\{\breve{T}_I, \breve{T}_D\right\} \tag{4.30}$$

It is important to mention that work policies pose constraints regarding $\breve{T}_D$. For instance, assuming the first four work policies, an activity *i* can only deliver its output to another activity *j* if *j* has been already received the input information of all chronological predecessors. Contrary, P$_5$ enables the flexibility for an activity *i* to neglect the output of a precedent activity *j* with k_j>0 and to start instead with its output at k_j=0.

After computing T_E, which is related to an activity *i*, the finish time of $T_{f_i(k_i)}$ is calculated. Thereby, all of the previously introduced effects due to crashing, partial rework, overlapping and learning must be considered. Subsequently, the next event T_E is computed changing simulation time again. This procedure is iterated until $\boldsymbol{H}$={} holds, i.e. no active activity is left (Figure 4.17).

5. Time-Cost Sensitivity Analysis of PD Processes Using Simulation

Prior to a time-cost tradeoff optimization, it is reasonable to initially understand the behavior of such a comprehensive model with respect to time and cost. This allows us not only to gain a better understanding of model dynamics but also to derive new managerial insights for some of the research gaps illustrated in Table 1.1. Therefore, we examined in this chapter the impact of work policies, process architecture, crashing and overlapping on process cost and schedule using simulation of 48 million artificially constructed processes. Notably, these four "high-level" model parameters can be directly controlled by process managers. In the first section, we will devote our attention to the effects of work policy and process architecture on process cost and duration. Subsequently, we separately analyze the efficiency of crashing and overlapping when applied to cyclic processes. This chapter concludes with a summary including the most important insights.

5.1 Time-Cost Investigation of Process Architecture and Work Policies

5.1.1 Introduction

When a new process must be established, process planners typically start with a macro perspective on the process and firstly define process architecture as well as work policy before they devote their attention to detailed parameters of individual activities such as crashing or overlapping (Figure 5.1). We followed this approach for our sensitivity analysis and hence start our investigation with the two "high-level" model parameters, namely process architecture and work policy.

According to definition 2.1, the architecture of a PD process defines not only the activities of a process but also the relationships between them as well as their temporal sequence of execution. Interestingly, studies to date have not yet derived scale-up behaviors for cost or duration of iterative processes due to changes of process architecture related parameters. Moreover, the influence of different work policies on the corresponding scale-ups has been neglected, too. To close these two research gaps is the objective of the following section.

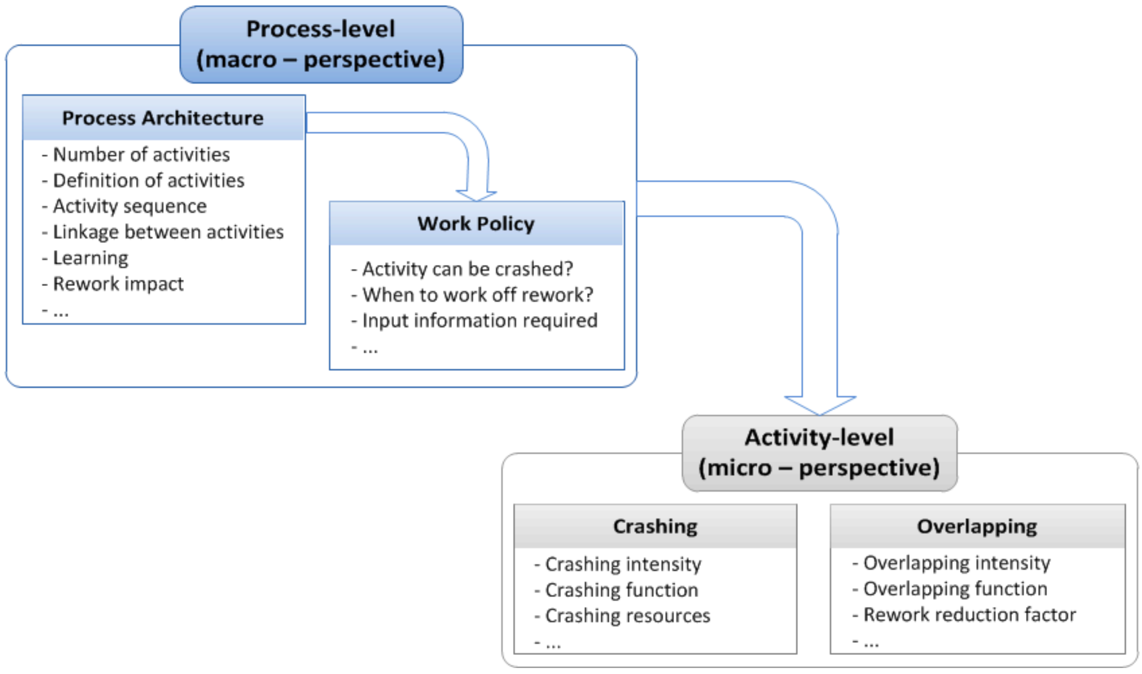


Figure 5.1: Typical approach for process planning.

5.1.2 Test setup

Model parameters

In general, our PD process model allows three distinct ways of modifying a process architecture:

1) Deletion/adding of activities in the process.
2) Deletion/adding/modification (i.e. changing the probabilities) of relationships between activities.
3) Rearrangement of the process sequence which really means changing the type of relationships between activities and thus influencing the information flow within the process. Importantly, the sequence does not affect coupling but only its implications (i.e. wait for input information or make assumptions).

Table 5.1 lists the parameters of our model which correspond to the above mentioned points and provides information whether a specific parameter was held constant throughout the upcoming test scenarios or variable. Additionally, a brief overview of the valuation for each parameter is provided.

Construction of artificial processes

To explore the impact of work policy and process architecture on time & cost we had to construct an adequate set of test processes featuring different characteristics. Thus, the model parameter M_1 appears to be variable in Table 5.1. Given a specified number of activities n_A for every test process, we count an exponential number[16] of different possibilities to assign relationships between the activities. Moreover, these relationships may feature different probabilities. Due to this apparent complexity, we had to find a trade-off between the number of process architectures simulated and generalization of simulation results.

[16] Exactly, $\sum_{i=1}^{n_A^2-n_A} \binom{n_A^2-n_A}{i}$ possibilities.

	Test Parameter			**Parameter Value**
		Constant	**Variable**	
Process Architecture	DSM - M_1		**X**	Artificially generated with random, sequential and mixed assignment of relationships.
	Number activities -n_A	**X**		20 activities for each artificial DSM M_1
	Number feedbacks - n_{fb}		**X**	Tests with 0,20,40 and 60 feedbacks within M_1 (randomly assigned)
	Number feed-forwards - n_{ff}		**X**	Tests with 30,60,90 and 120 feed-forwards within M_1 (randomly assigned)
	Feedback probabilities $m_{1,ij}$		**X**	Tests with $m_{1,ij}$=0.5, $m_{1,ij}$=0.3, $m_{1,ij}$=0.1
	Process Sequence - $\boldsymbol{S}$		**X**	Tests with random permutations of $\boldsymbol{S}$
	Activity cost - c_i	**X**		Cost for each activity with $c_i \in [9,11]$
	Activity duration - t_i	**X**		Duration for each activity with $t_i \in [9,11]$
	Learning and rework impact - λ_{ij}		**X**	Tests with λ_{ij}=0.1, λ_{ij} =0.3 and λ_{ij}=0.5
	Max. iterations caused by any other activity - $\hat{k}_i$		**X**	Tests with $\hat{k}_i = 5$, $\hat{k}_i = 10$ and $\hat{k}_i = 15$

Table 5.1: Process architecture related model parameters for the sensitivity analysis.

As a consequence, we set n_A=20 constant for each test process. This number is adequate to obtain qualitatively usable results which also reveal the differences between different work policies or architecture related strategies. Holding this variable constant also allows for greater understanding of other variables. Each activity was assigned a single deterministic value for cost and duration with $c_i / t_i \in [9,11]$ (see Table 5.2) as this determinism makes comparisons of different test scenarios less noisy.

To cut the vast number of different relationship assignments, we constructed processes exhibiting different numbers of feed-forward relationships, n_{ff}, and feedback relationships, n_{fb},

c_1/t_1=9.7	c_5/t_5=10.0	c_9/t_9=9.4	c_{13}/t_{13}=9.8	c_{17}/t_{17}=10.7
c_2/t_2=10.3	c_6/t_6=9.1	c_{10}/t_{10}=9.9	c_{14}/t_{14}=10.8	c_{18}/t_{18}=9.7
c_3/t_3=10.9	c_7/t_7=10.2	c_{11}/t_{11}=9.5	c_{15}/t_{15}=9.6	c_{19}/t_{19}=10.0
c_4/t_4=10.7	c_8/t_8=10.6	c_{12}/t_{12}=10.2	c_{16}/t_{16}=9.1	c_{20}/t_{20}=10.9

Table 5.2: Cost and duration values for each activity of the artificial DSMs.

as follows. Firstly, we set $n_{ff} = \{30, 60, 90, 120\}$ since this set of values corresponds to a realistic "relationship density" of 15-60% of all possible feed-forward relationships.[17] Contrary, we defined $n_{ff} = \{0, 20, 40, 60\}$ as feedback relationships are supposed to occur less likely than feed-forward ones in most real world PD processes. So, for our tests, we obtain 16 different n_{ff}/n_{fb} combinations.

For any given n_{ff} and n_{fb} combination, we constructed 1.000.000 distinct processes as follows (see Figure 5.2). Firstly, we randomly allocated all feed-forward relationships to the 20 activities. Then, the underlying number of feedbacks was randomly assigned to the activities 1.000 times while feed-forward assignments remained constant. In this way, we yielded 1.000 test processes. Afterwards, we randomly determined a new assignment for the feed-forward relationships subsequently adding feedbacks, again 1.000 times. This procedure is iterated 1.000 times consequently providing 1.000.000 test processes for a single n_{ff}/n_{fb} combination. Due to the 16 n_{ff}-/n_{fb} combinations we finally gain 16.000.000 random test processes. Notably, an activity must not necessarily be connected with another because of the random assignments.

Beside these completely random process architectures, we also constructed two other process types: 1.) *sequential processes* and 2.) *mixed processes*. For their generation, we used the same construction procedure as in case of random processes hence generating 16.000.000 distinct process instances for each process type. Though, regarding sequential processes, we

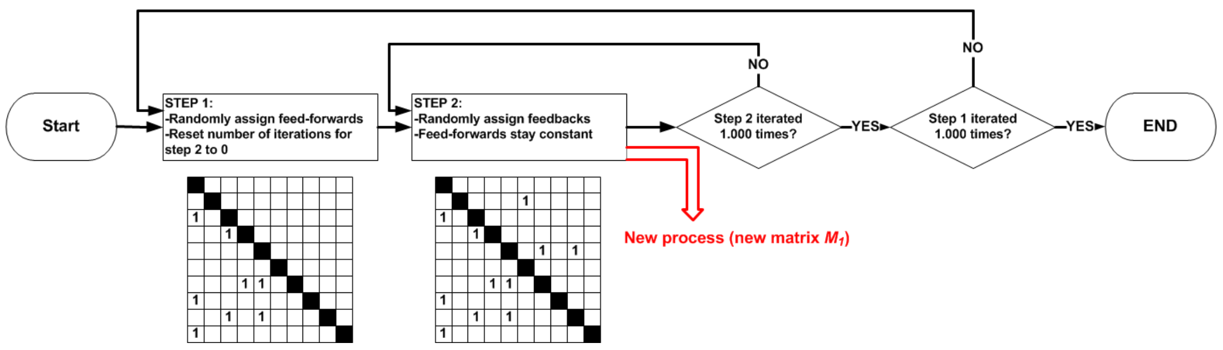

Start
STEP 1:
-Randomly assign feed-forwards
-Reset number of iterations for step 2 to 0
STEP 2:
-Randomly assign feedbacks
-Feed-forwards stay constant
Step 2 iterated 1.000 times?
NO
YES
Step 1 iterated 1.000 times?
NO
YES
END
New process (new matrix M_1)

Figure 5.2: Illustration of constructing artificial DSMs.

[17] Assuming 20 process activities we obtain 190 possible feed-forward/feedback relationships.

ensured during the assignment of feed-forward relationships that adjacent activities are always sequentially linked in order to guarantee strictly sequential information flows within the processes. In contrast, mixed processes exhibit sequential information flows combined with parallel activities. For this purpose, we merely connected every second activity with its adjacent activity. Figure 5.3 depicts two process architectures for each of the three different

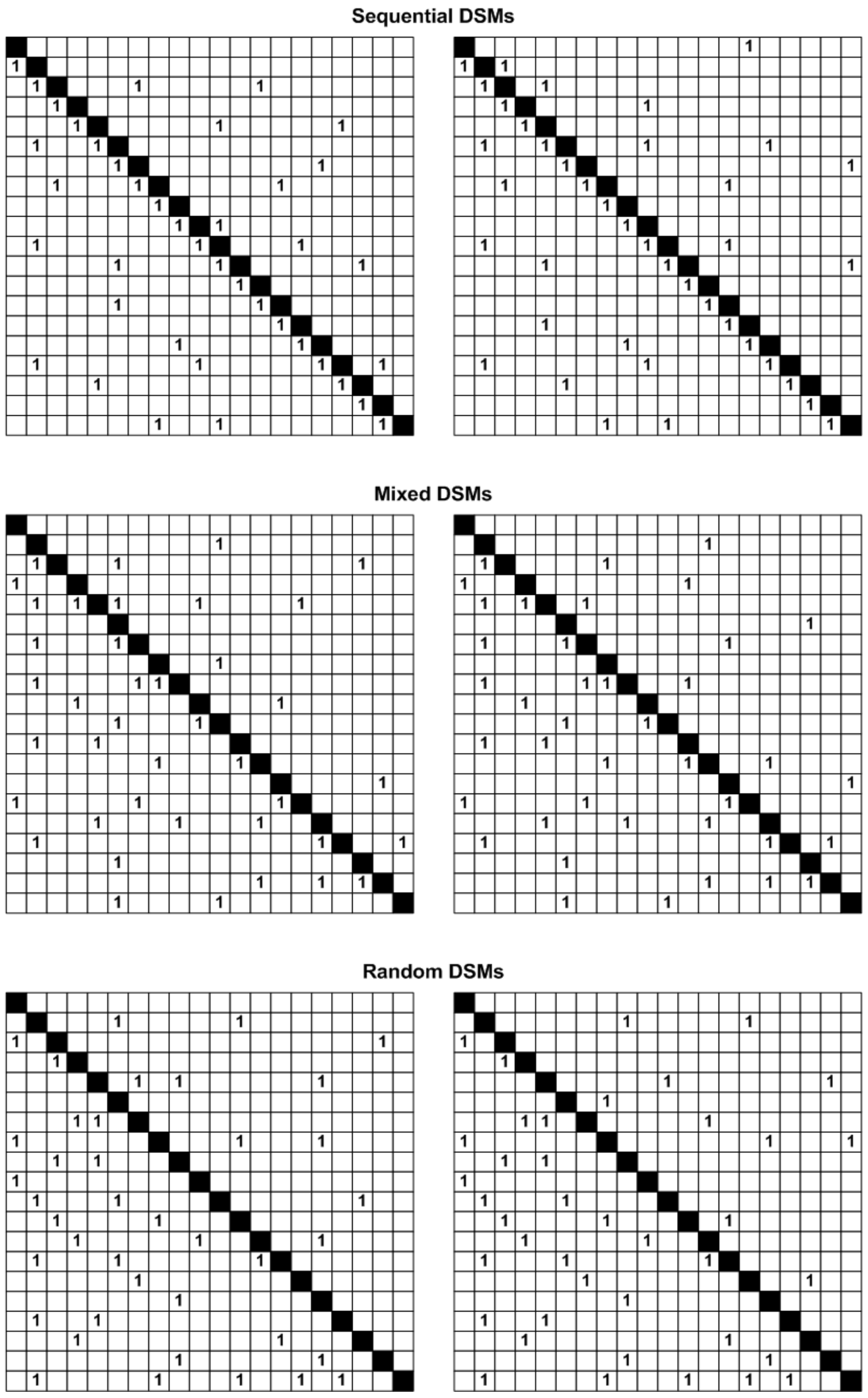


Figure 5.3: Exemplary artificial process architectures.

process types. According to the just outlined process construction, random processes inherently feature most parallelism between activities, followed by mixed processes while sequential test processes are characterized by the least parallelism.

5.1.3 Scale-up behavior for time and cost

5.1.3.1 Impact of relationships

Neglecting the effects of overlapping and crashing, feedbacks in a process will generally augment cost and duration associated to an individual activity. Though, processes constitute systems with activities (i.e. system elements) interacting among each other. Therefore, it is usually not obvious how cost and duration of the entire process (=system) are changed through feedbacks. Besides, work policy rules set up constraints concerning the processing of activities which is supposed to impact c_{tot} and t_{tot} as well. Thus, we were interested how both, process architecture related parameters and work policy, affect the scale-up behavior of process cost and duration. With respect to work policies, we resigned on a consideration of P_3 in this section as it differs from P_2 only in terms of crashing and overlapping while work/rework must be processed in the same ways as regulated by P_2. However, the entire process architecture related model parameters listed in Table 5.1 were considered. In the underlying subsection (which corresponds to a base case), we mainly focused on the impact of relationships and therefore treated the following subset of parameters constant:

- Process Sequence ***S***
- Learning and rework impact: λ_{ij}=0.5
- Maximal number of iterations for activity *i* caused by any other activity: $\hat{k}_i = 5$

Impact on cost

Applying the discrete event simulation (as described in section 4.7) to the random test processes for different n_{ff}/n_{fb} combinations assuming work policy P_1, P_2, P_3 and P_4 yields the average cost values displayed at Figure 5.4. While absolute values differ for the four work policies we instantly note that, regardless of work policy, cost rather increase logarithmically, depending on the number of relationships, than exponentially. In fact, iterations make every activity pricier. But rework impact and learning curve effects (like defined by equation 4.1) absorb rework cost thereby preventing its exponential boost. Importantly, the scale-up behavior is not explicitly governed by n_{fb} but also by the number of feed-forwards since absolute process cost apparently rise with increasing n_{ff} for identical n_{fb} (assuming n_{fb}>0). Figure A.5 in the appendix visualizes the respective scale-up as a function of n_{ff} and the

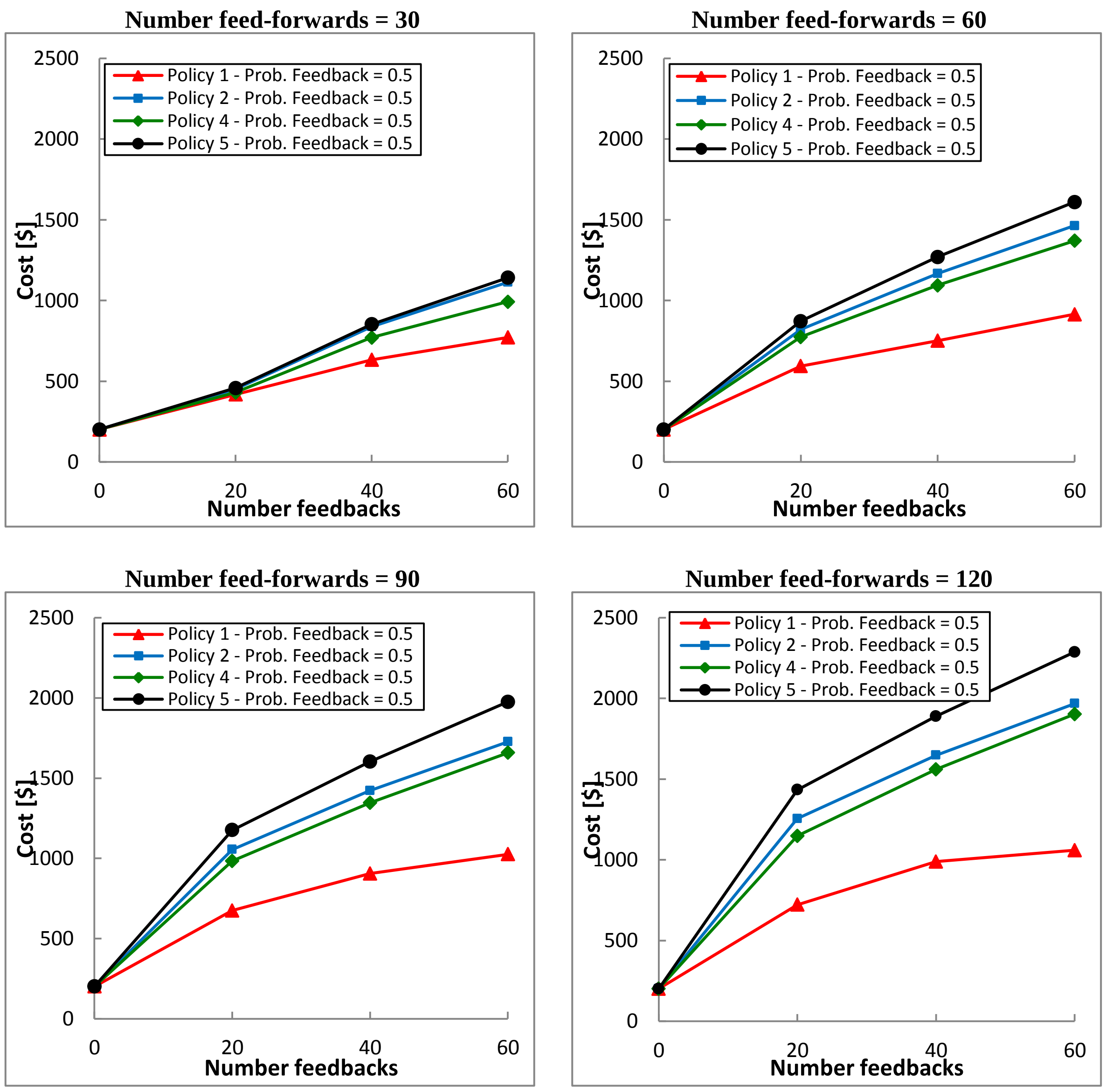


Figure 5.4: Comparison of cost outcomes for work policies P_1, P_2, P_4 and P_5 assuming random processes and a feedback probability 0.5.

results point out a linear growth rate of cost for all work policies: $\boldsymbol{O}(n_{ff})$. Contrary, we claim that the number of feedbacks accumulates process cost logarithmically: $\boldsymbol{O}(\log(n_{fb}))$. Moreover, both growth rates hold for sequential and mixed processes, too. The corresponding charts, depicted at Figure A.1 and Figure A.2, can be found in the appendix. Nevertheless, regarding absolute cost values, we note – to some extent serious – discrepancies between the applied work policies. For instance, P_5 produced in average twice as cost-intensive processes as P_1 assuming n_{ff} = 120 and n_{fb}=60.

Usually, increasing the probability of feedback within a process is supposed to result in

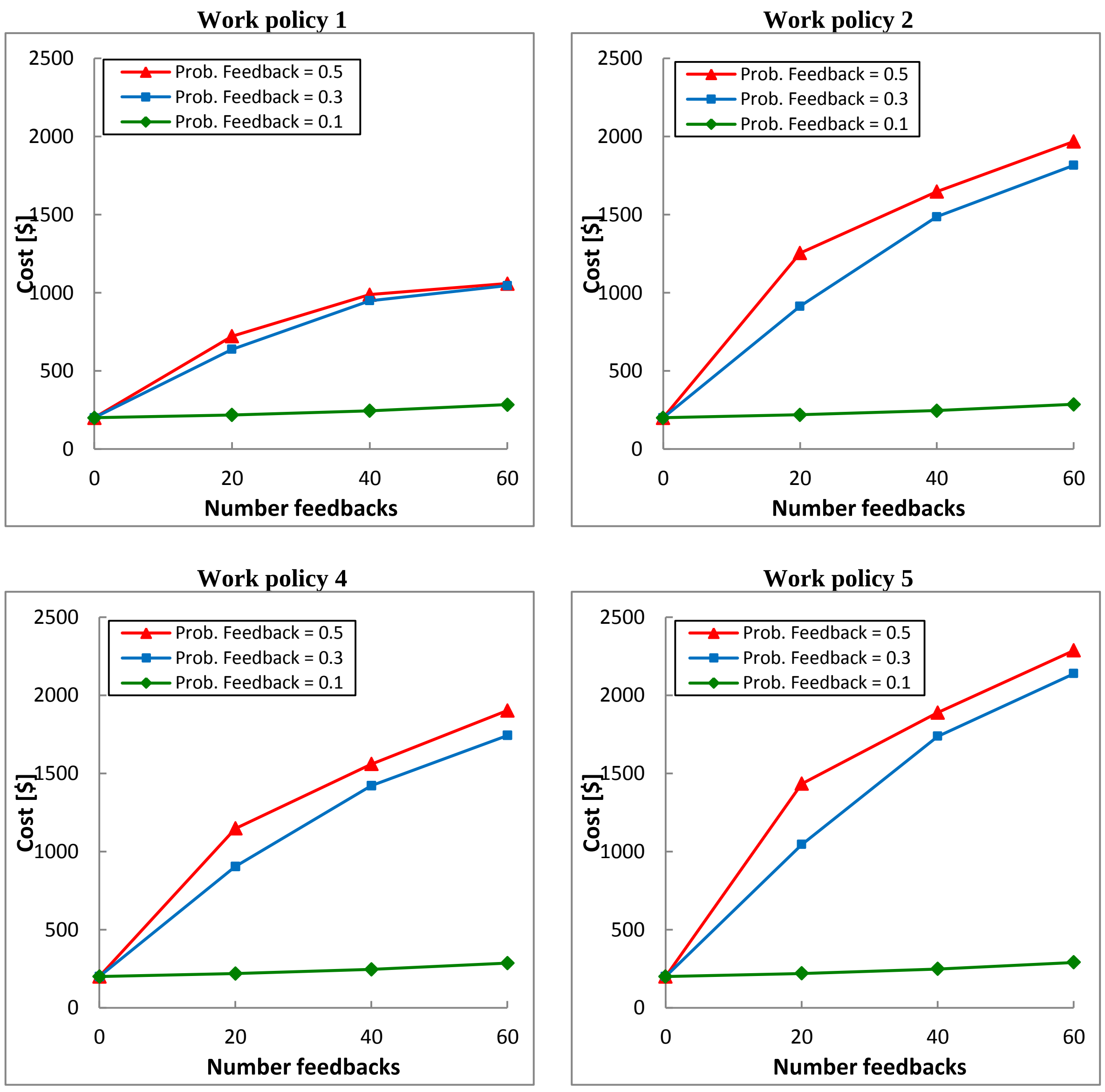
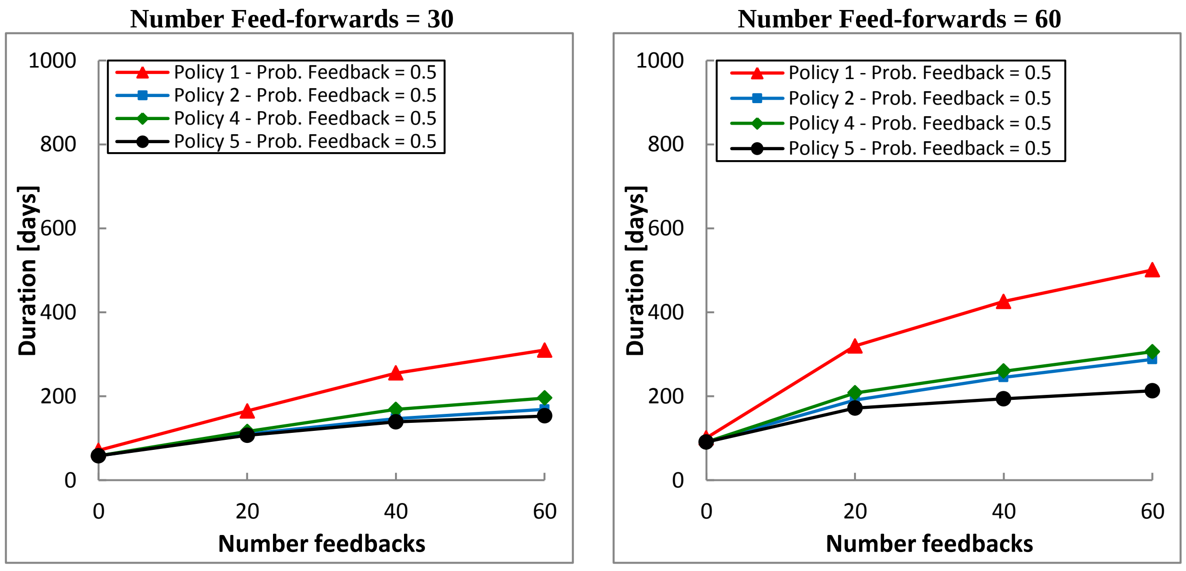


Figure 5.5: Comparison of cost outcomes for work policies P_1, P_2, P_4 and P_5 assuming random processes with 120 feed-forwards and different feedback probabilities.

more rework and ultimately higher cost & duration. Yet, this downsides may be diluted through constraints such as the maximal number of iterations allowed, $\hat{k}_i$, or an exponential reduction of rework cost due to learning and rework impact (see equations 4.1 and 4.2). For instance, the relationships of a process could feature a (theoretical) feedback probability of 100% and, still, overall cost and schedule might not significantly differ compared to a feedback probability of 30%. Figure 5.5 addresses this issue and shows that cost curves for feedback probabilities 0.5 and 0.3 vary only marginally for all four investigated work policies.

Impact on duration

In contrast to cost, process lead-time is not necessarily a summation of the individual activities' durations, mainly because of potential parallelisms between activities. Hence, feedbacks in the process could leave the critical path, and thus overall process lead-time, unchanged. Nevertheless, the average growth of duration for the different test process types (results for random processes are plotted in Figure 5.6, for other types in appendix Figure A.3 and Figure A.4) in dependence of n_{fb} appears to be very similar to that for cost. Also, an increasing number of feed-forwards (assuming n_{fb}>0) provokes a linear scale-up of process time (Figure A.6), just like for cost. Notably, work policies responsible for inexpensive



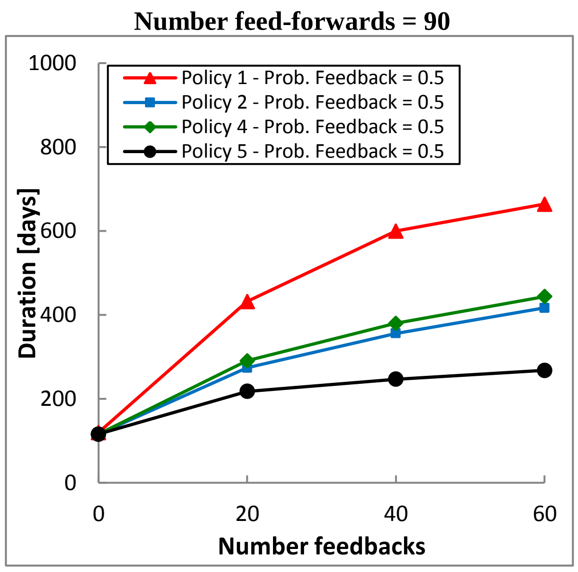
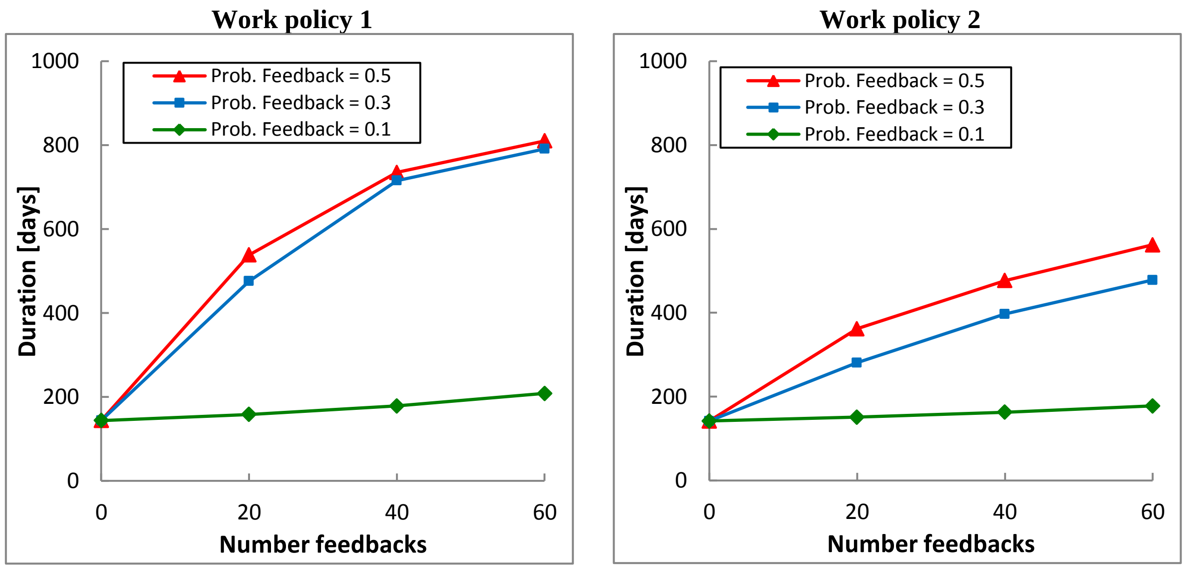


Figure 5.6: Comparison of schedule outcomes for work policies P_1, P_2, P_4 and P_5 assuming random processes and a feedback probability 0.5.

processes, e.g. P_1, seem to produce processes with high duration. We will investigate this issue more in detail in subsection 5.1.4. Finally, Figure 5.7 shows the impact of feedback probability on process duration for all work policies.

Min-Max values

It is important to allude that a great variance regarding minimal and maximal cost/schedule values for a specific n_{fb}/n_{ff} combination of the simulated processes exists. Exemplarily,



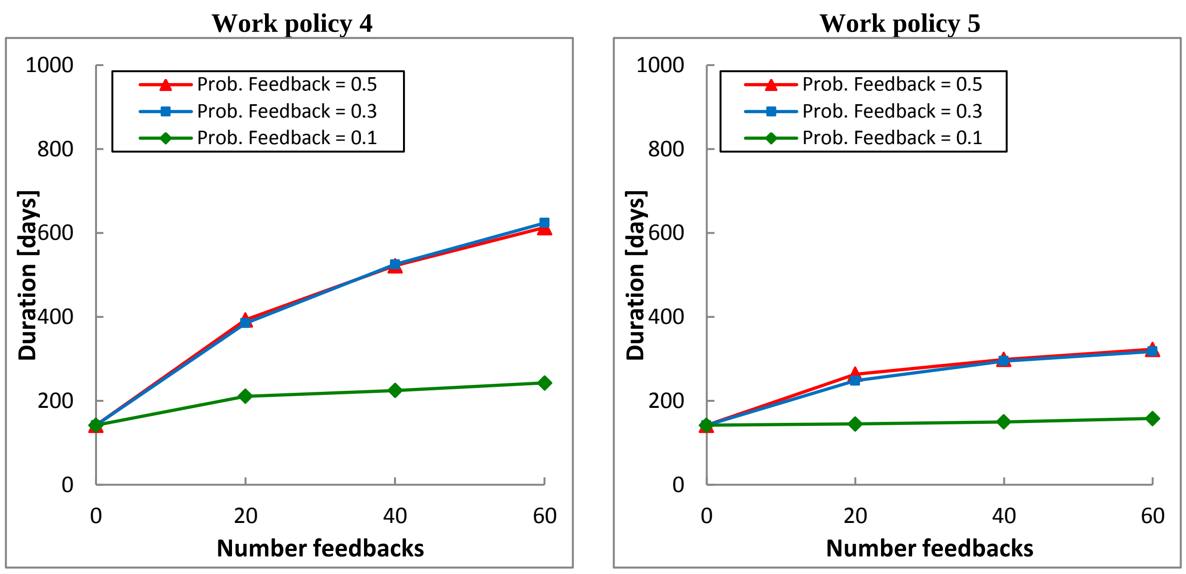


Figure 5.7: Comparison of schedule outcomes for work policies P_1, P_2, P_4 and P_5 assuming random processes and 120 feed-forwards.

Figure 5.8 illustrates minimal and maximal cost/schedule results for random processes with 120 feed-forwards assuming a feedback probability of 0.5. The mentioned variance for both, cost and duration is evident and proved that a reduction of n_{fb} does not necessarily result in a reduction of process cost or time. Referring to Figure 5.8, a process architecture with n_{fb}=20 could actually cause cost as high as for a process featuring n_{fb}=60. Consequently, objective functions for process sequencing which merely rely on a minimization of n_{fb} (e.g. Steward

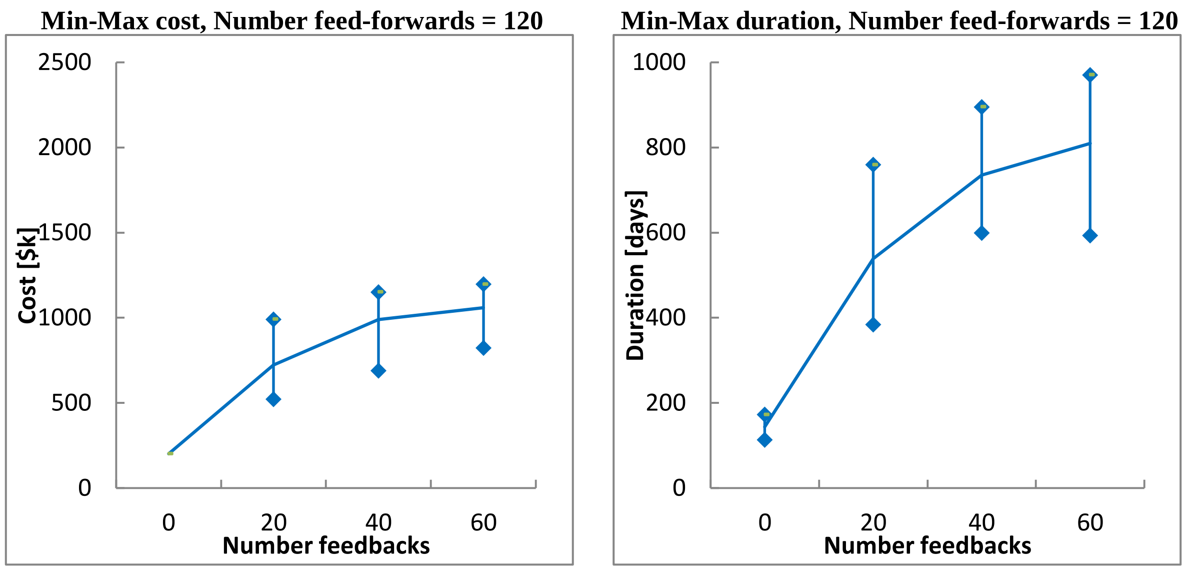


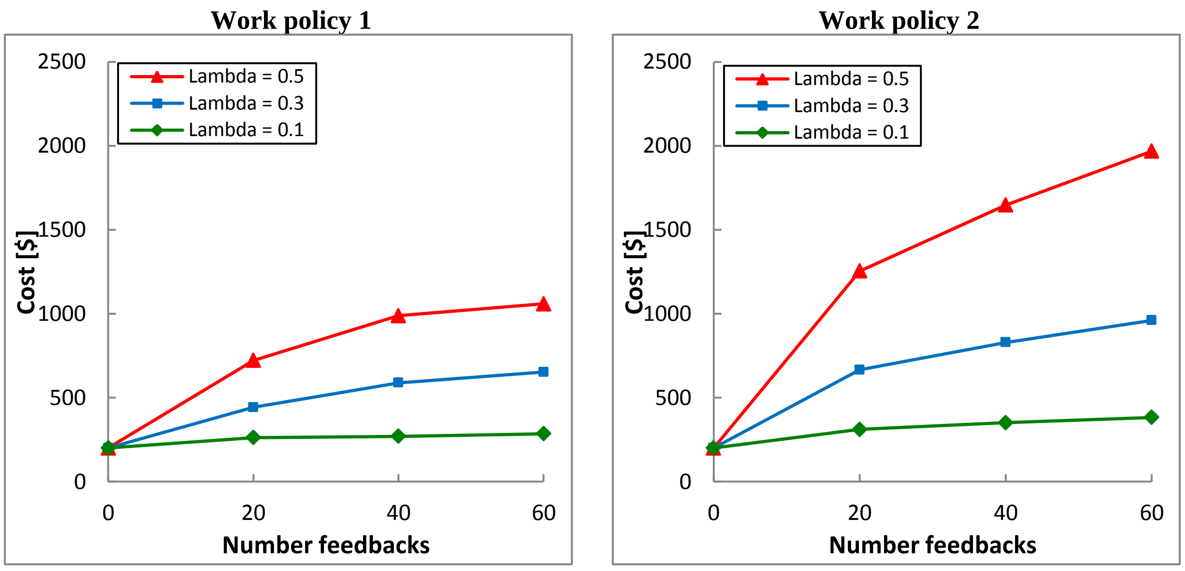
Figure 5.8: Min-Max values for cost and duration of P_1 assuming random processes, 120 feed-forward relationships and a feedback probability 0.5.

1981, Gebala and Eppinger 1991, Kusiak and Wang 1993) may indeed result in stable processes with less perturbation through feedbacks. Though, this may not be the cheapest and/or most expeditious. Hence, we suggest resigning on such deterministic proxies and instead using simulation techniques to determine process cost and duration.

5.1.3.2 Impact of learning, rework impact and activity sequence

Effects of λ

The previous test results clearly pointed out that a low number of feedbacks does not guarantee a reduction of overall process cost c_{tot} or duration t_{tot}. A reason for this outcome is the fact that, theoretically, a process might contain numerous feedback relationships provoking, however, only minor cost/duration for rework if rework impact and learning curve values are low.[18] Reversely, few feedbacks within a process could cause the unfavorable repetition of non-robust activities ultimately ending in high process cost and duration. In order to empirically demonstrate this proposition on the one hand and to determine the actual impact on cost and duration through changes in λ_{ij} we conducted a series of tests, partially depicted at Figure 5.9 and Figure 5.10 for random processes (the other test results can be



[18] At this point, we like to repeat that low values in the learning curve vector indicate that learning is high (see section 4.2).

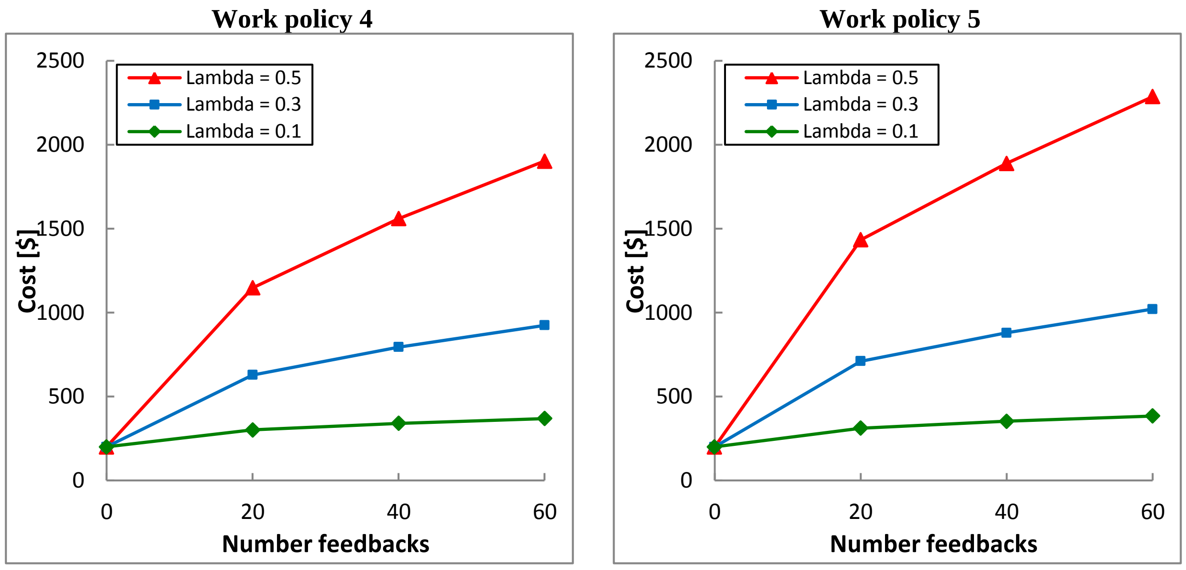


Figure 5.9: Cost comparison of lambda values for work policies P_1, P_2, P_4 and P_5 assuming random processes with 120 feed-forwards and a feedback probability 0.5.

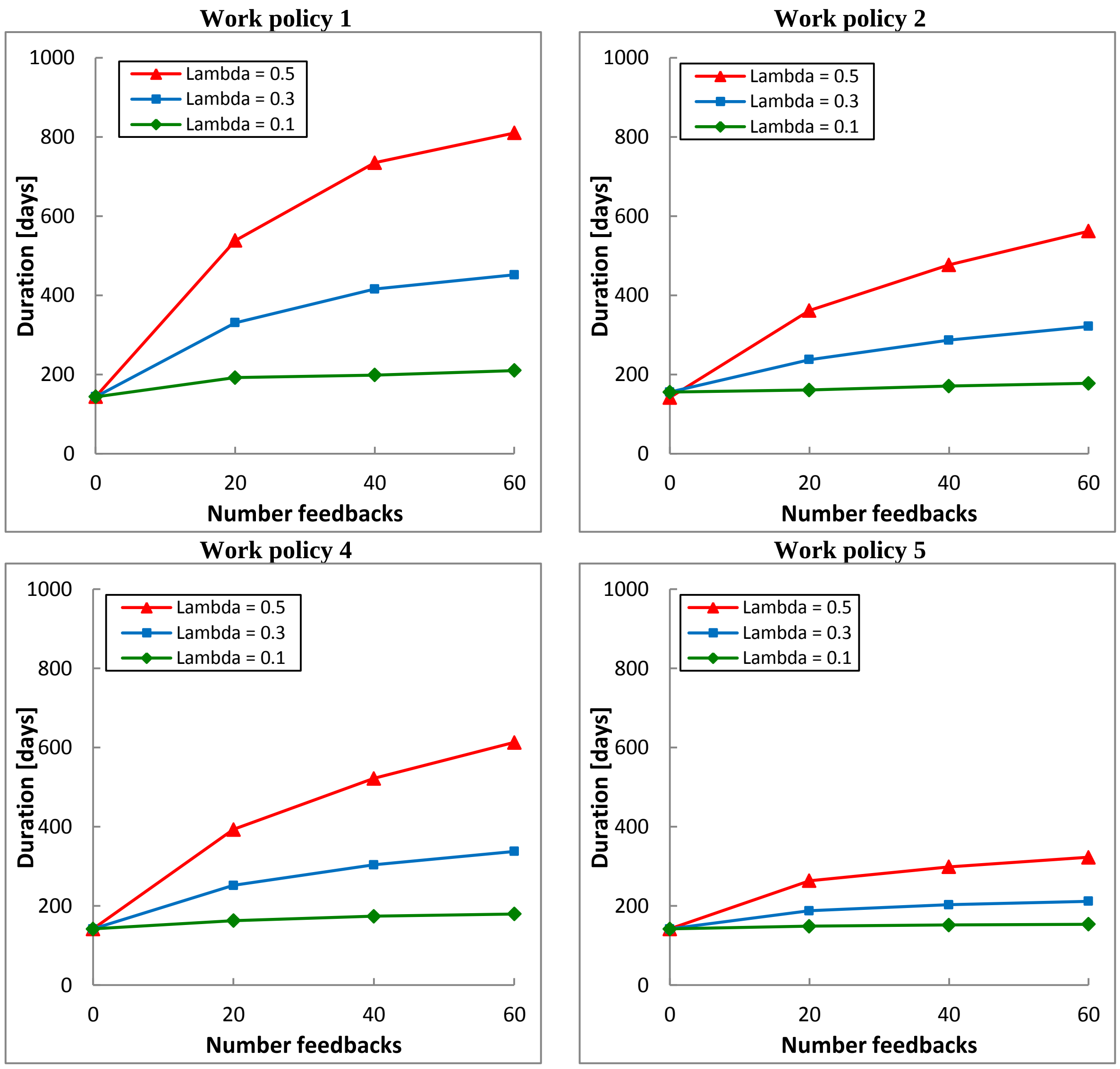


Figure 5.10: Schedule comparison of lambda values for work policies P_1, P_2, P_4 and P_5 assuming random processes with 120 feed-forwards and a feedback probability 0.5.

found in the appendix). Studying both figures reveals that modifying λ_{ij} does not affect the scale-up behaviors we noted before. Instead, λ_{ij} seems to merely act as a scaling factor for both, process cost and schedule.

Maximal number of iterations

Beside λ_{ij}, the maximal number of repetitions allowed for an activity *i* provoked by any other activity, namely $\hat{k}_i$, constitutes another model parameter concerning learning. Therefore, we examined the sensitivity of $\hat{k}_i$ on cost and duration by simulating the artificial test processes of all three process types for $\hat{k}_i = \{5,10,15\}$. Intuitively, we would expect a steeper cost/duration increase with growing number of iterations. But instead, all cost/schedule curves appear to overlap for any $\hat{k}_i$ (e.g. Figure A.7 in the appendix). This observation holds for all work policies since rework cost/duration decline exponentially because of learning and rework impact ultimately resulting in a convergence of λ_{ij} close to zero even after a few iterations.

Constant vs. exponential learning

In a subsequent test bank, we therefore intentionally applied a constant λ_{ij} in any iteration to its original cost or duration (i.e. c_i and t_i). We refer to this strategy as “constant learning” (CL) since λ_{ij} remains constant as opposed to the previous strategy where λ_{ij} decreases exponentially (“exponential learning” - EL). Thus, the following equations hold for CL:

$$\tilde{c}_i(k_i) = c_i \cdot \lambda_{ij} \text{ , } \forall k_i \tag{5.1}$$

$$\tilde{t}_i(k_i) = t_i \cdot \lambda_{ij} \text{ , } \forall k_i \tag{5.2}$$

Because of the constant value for $\tilde{c}_i(k_i)$ or $\tilde{t}_i(k_i)$ in higher order iterations, the CL strategy is supposed to result in higher rework cost compared to the EL approach. But with increasing number of relationships cost/duration growth might be restricted through $\hat{k}_i$ as the growth of cost and duration can only continue if activities are allowed to be reworked. Figure 5.11 displays for the investigated work policies the scale-up behaviors for three of the previously described cases assuming random processes with 120 feed-forwards: 1) $\hat{k}_i = 15$ for exponential learning 2) $\hat{k}_i = 15$ for constant learning and 3) $\hat{k}_i = \infty$ for constant learning. Noteworthy, feedback probability was set to 0.2 since some of the simulation runs for $\hat{k}_i = \infty$ required enormous computational time for high feedback probabilities. As a result, we

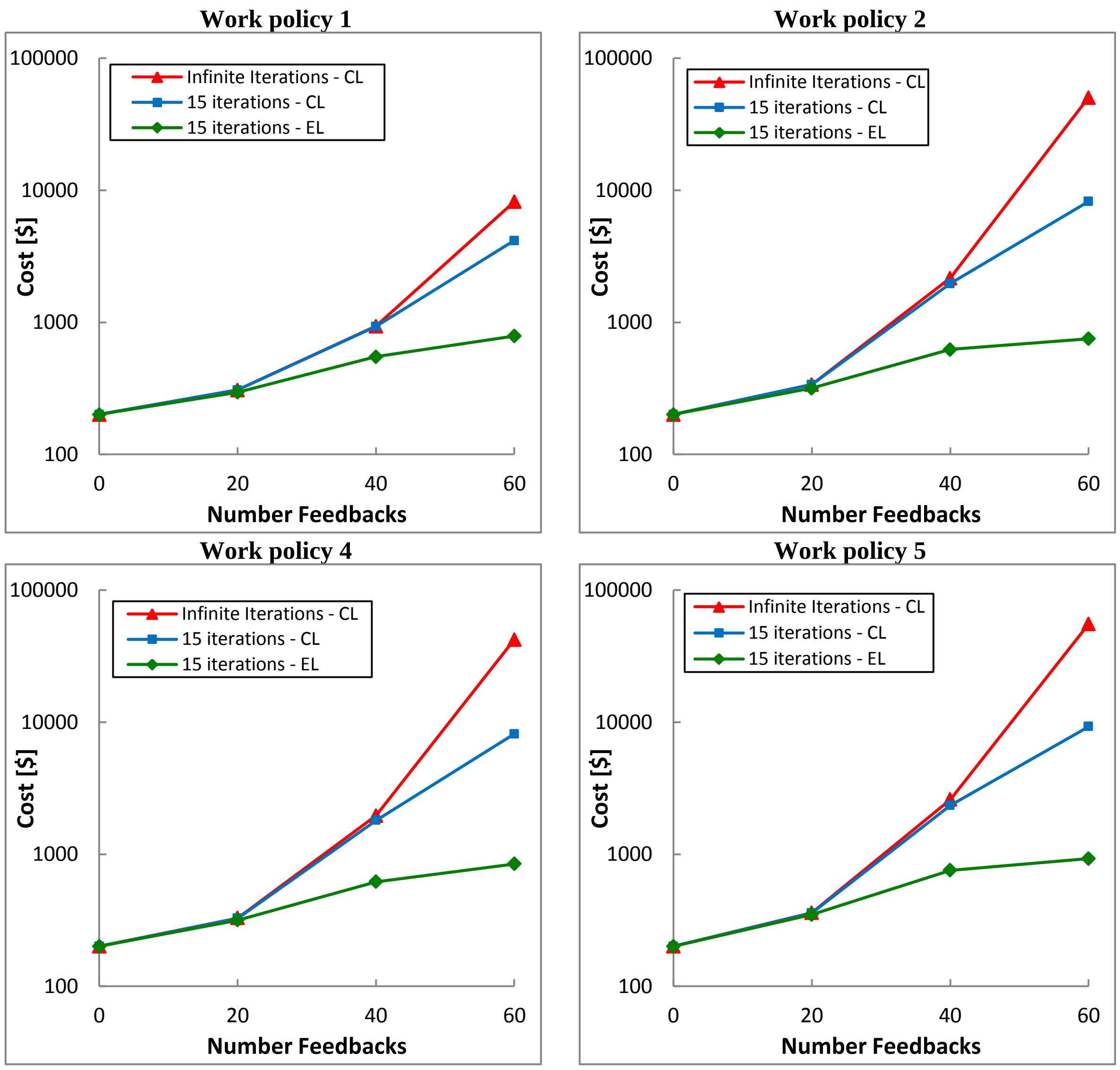


Figure 5.11: Cost comparison of learning strategies for work policies P_1, P_2, P_4 and P_5 assuming random processes with 120 feed-forwards and a feedback probability 0.2.

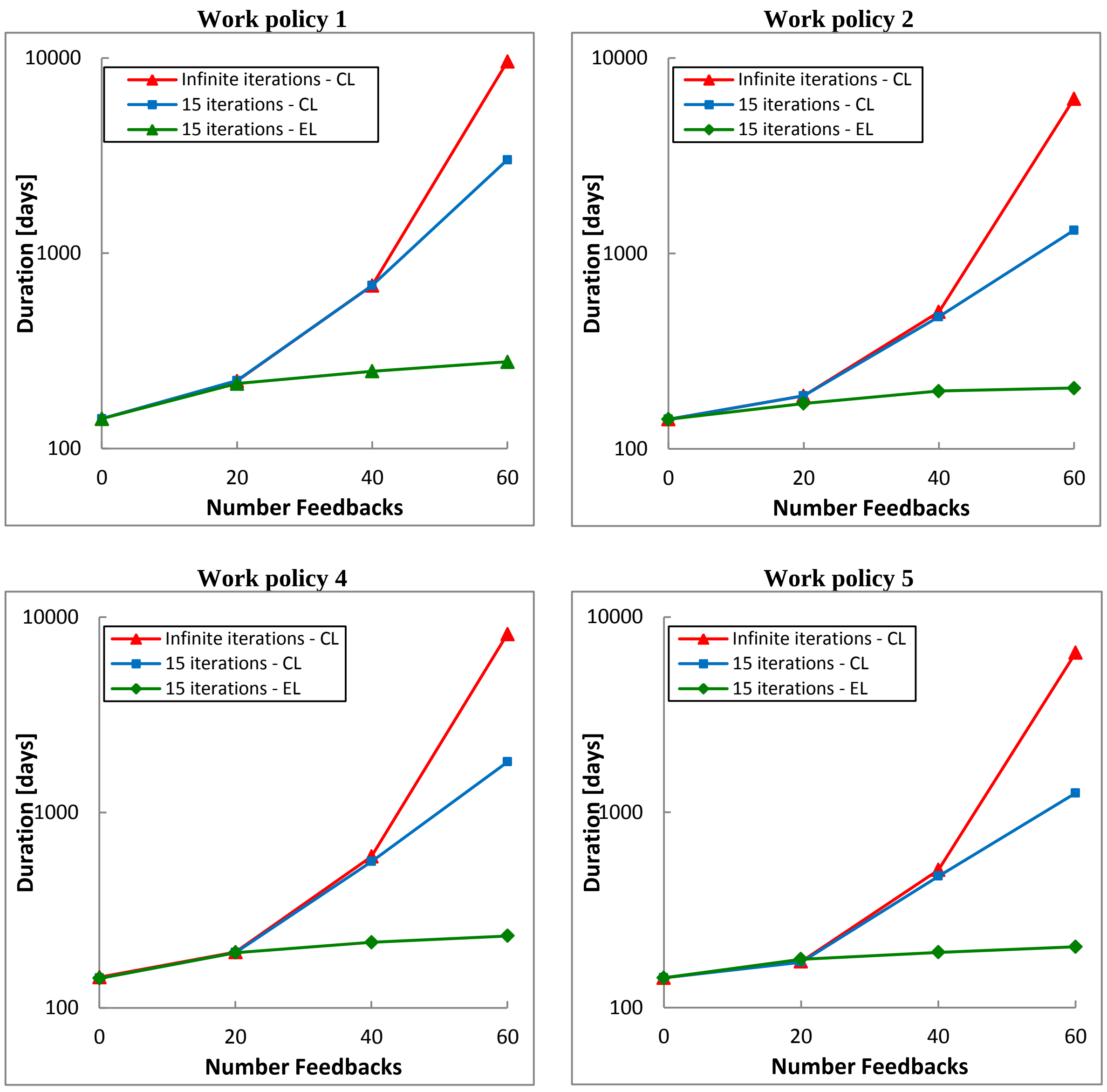


Figure 5.12: Cost comparison of learning strategies for work policies P_1, P_2, P_4 and P_5 assuming random processes with 120 feed-forwards and a feedback probability 0.5.

empirically confirmed that $\hat{k}_i$ is not a crucial model parameter as long as we assume an exponential learning. In fact, $\hat{k}_i$ will only become a decisive model parameter with respect to c_{tot} and t_{tot} if we alter the initial calculation of λ_{ij} according to equations 5.1 and 5.2. Then, scale-up behavior will shift from a logarithmic to an exponential one.

Impact of activity sequence

Altering the activity sequence may decisively influence the outcome of process cost and schedule. It is must be mentioned that changing the activity sequence does not vary the quantity of relationships or the activities associated with a certain relationship. Instead, the activity sequence determines the starting dates of activities through potentially transforming sequential feed-forward relationships to feedbacks and vice versa. In order demonstrate the effect of activity sequencing, we just picked any of the randomly generated processes with n_{ff}= 60 and n_{fb}=20. Subsequently, we created 5000 random (and distinct) activity sequences out of the 20! possibilities and simulated them. The corresponding cost and duration results are plotted at Figure 5.13. Clearly, a strong variation in cost is noticeable, ranging from \$407 to maximal \$724 with an average value of \$574. Similar, duration values vary from 186 days to 326 days with an average of 256 days. Apparently, the monetary consequences can be enormous if managers select a suboptimal process sequence.

5.1.4 Time-Cost tradeoffs due to work policy

Analyzing Figure 5.4 and Figure 5.6 already indicated potential time-cost tradeoff decisions due to the choice of a work policy. Thus, we assume that time-cost tradeoffs do not only emerge from crashing/overlapping decision or from process architecture related parameters like the activity sequence, but also from the choice of the underlying work policy. In order to examine this proposition more in detail, we approximated the time-cost objective space of random processes with 120 feed-forwards for P_1, P_2, P_4 and P_5 (for the other two process

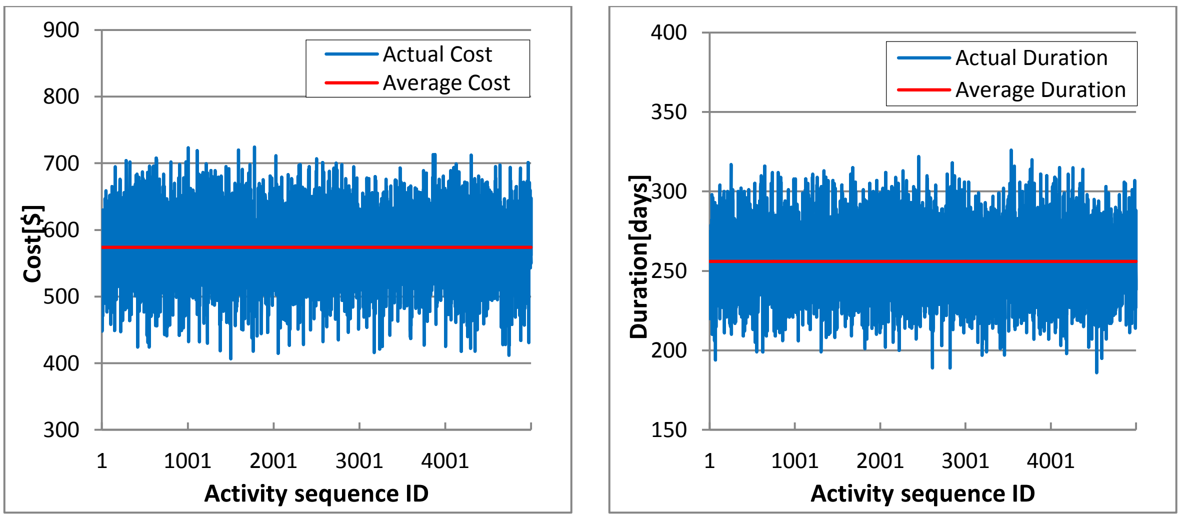


Figure 5.13: Cost and duration for different activity sequences.

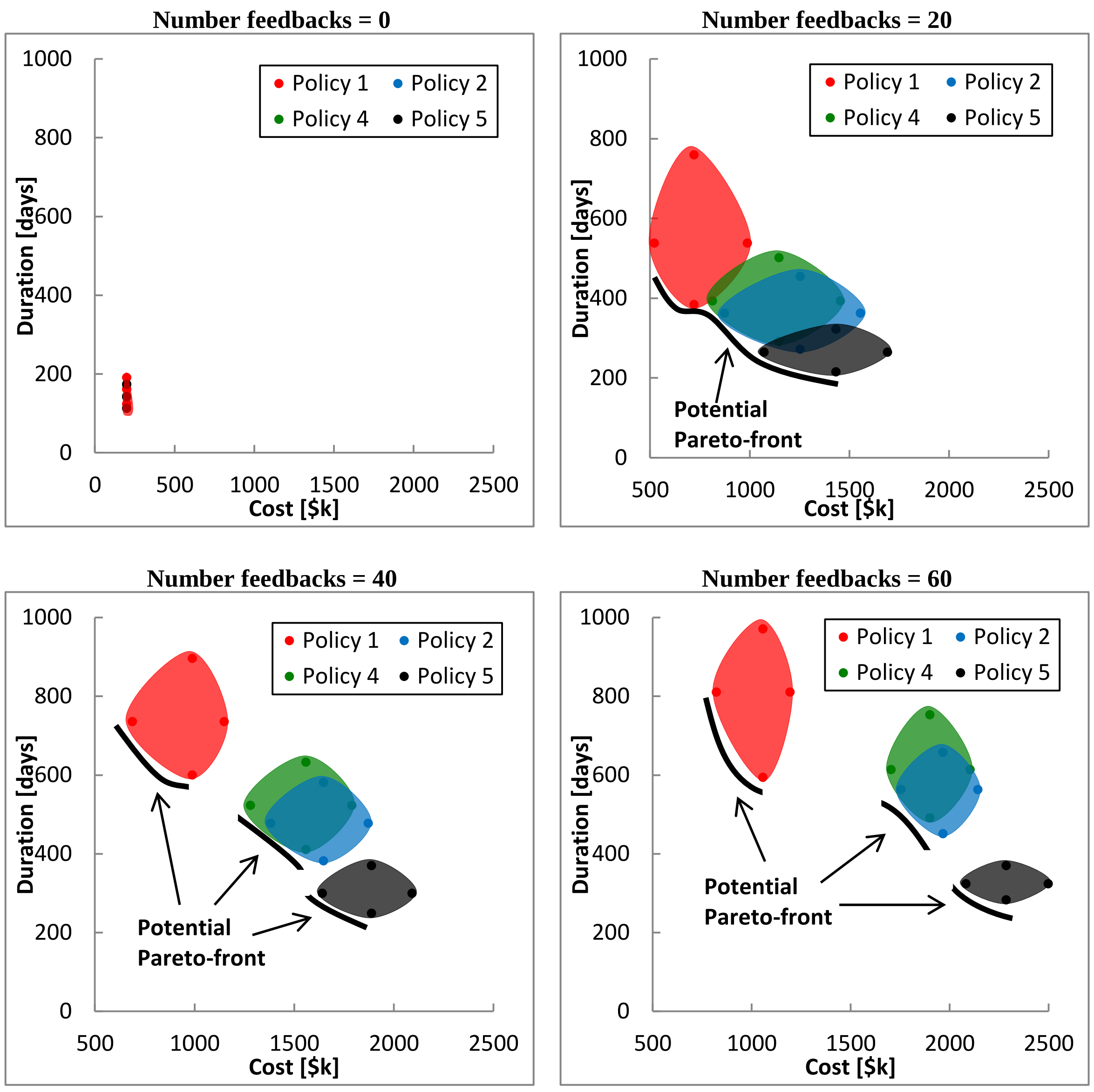


Figure 5.14: Time-cost approximation of objective spaces for policies P_1, P_2, P_4 and P_5 assuming random processes with 120 feed-forwards and 0.5 feedback probability.

types the reader may refer to Figure B.8 and Figure B.9 in the appendix) with respect to different n_{ff} (see Figure 5.14). For this purpose, we used the simulated min-max cost & duration values as four corners of the space and approximated the shape between these corner points. The resulting objective spaces highlight the suggested trade-offs. Apparently, the discovered tradeoffs would not occur for acyclic processes like visualized by the case n_{fb}=0 in Figure 5.14. In this example, policy 1 is dominated by the other policies since costs are equal for all policies but duration is highest for policy 1. The apparent existence of a Pareto-front itself furthermore proves the definition of our four/five work policies to be reasonable as the rules of every work policy involve advantages and disadvantages regarding cost and schedule. As a general observation, the dynamics of feedbacks amplify the time-cost differences between our defined work policies since the corresponding time-cost Pareto-front spreads out more with increasing number feedbacks. Also, the approximation of the work policy objective spaces illuminates different volatilities in terms of cost and duration. For instance, while P_1 seems to be more volatile with respect to duration, P_5 tends to be very robust concerning schedule but very volatile with respect to cost.

Regarding the shape of a potential Pareto-front for all four work policies, the approximated objective spaces indicate that it must not necessarily be coherent. Instead, it appears that the Pareto-front features discontinuities, mainly with respect to process cost, with increasing amount of feedbacks. This observation is counterintuitive as we would assume that more process dynamics provoke larger objective spaces hence leading to a merging of the objective spaces (with continuous Pareto-front) in lieu of a spread-out. But as explained earlier, learning curve & rework impacts as well as a maximal limit for the allowed number of iterations prevent a higher volatility of the objective spaces with more feedbacks. Besides, with more feedbacks – and thus greater rework cost – the cost (and resource) conserving work policy rule to prohibit non-adjacent activities to work in parallel (see Table 4.1) seems to become more dominating. This rule is only part of P_1 and obviously generates processes in a cost region which cannot be attained by the other work policies.

While a work policy may be superior to all other work policies for certain processes and process parameters, the existence of a “super work policy” dominating any other policy for all possible processes is very unlikely. Schedule related advantages of a work policy are typically realized through the promotion of parallelism within the process - but at the expense of additional cost for rework (e.g. P_5 fosters iterative overlapping but generates very costly processes). Vice versa, cheap processes are characterized by work policy rules which prevent the occurrence of (expensive) rework, and thus a more sequentially linked process with low amount of iterative overlapping /parallelism. Consequently, a “super work policy” had to strongly foster parallelism within the process without provoking additional cost. Indeed, such a scenario might be possible for a process with high learning & low rework impact rates but rather as a result of process specific parameters and not as a result of work policy rules.

5.1.5 Summary

The previous time-cost investigation of process architecture and work policies using simulation revealed interesting insights and propositions which can be summarized in the following propositions:

1. Regardless of work policy, scale-ups of cost and duration for cyclic processes significantly depend on the definition of learning and rework impact, λ_{ij}. If rework cost/schedule decline exponentially due to λ_{ij}, process cost and duration grow logarithmically with the number of relationships in the process. Contrary, the scale-up becomes exponential if λ_{ij} occurs at a constant rate and, in addition, the number of maximal iterations is high (preferably not limited by an upper bound). From a practical perspective, the logarithmical scale-up behaviors are interesting as they suggest that product quality, which is influenced by the number of iterations, could be strongly enhanced at a more or less moderate increase of overall cost and duration.
2. Feedback probabilities, the value of λ_{ij} itself, or the maximal number of iterations separately rather influence absolute cost/duration values than general growth rate.
3. Assuming identical test processes, the four/five work policies defined in chapter 4 cause different outcomes for process cost and duration. Interestingly, no work policy appeared to be superior to the others in both (cost, duration) dimensions. Instead, a time-cost tradeoff due to the choice of work policy arises. Therefore we suggest extending the time-cost tradeoff problem by a further managerial lever (beside crashing, overlapping and process architecture), namely work policy.

5.2 Time-Cost Investigation of Crashing

5.2.1 Introduction

While the immediate effects of crashing an individual activity regarding its time and costs have been already explained in section 4.3, the upcoming sensitivity analysis addresses the influence of crashing on process cost & time assuming cyclic processes. In particular we were interested how modifications of cyclic process architectures and the choice of work policy influence economic efficiency of crashing. As illustrated by Table 1.1, previous literature on crashing merely considered acyclic process models for similar analysis and neglected the potential influences of work policy, too.

Crashing for cyclic processes

According to the definition of crashing intensity and crashing function for a single activity *i* we immediately know whether it was economic reasonable to crash *i* or not. If in any iteration

$$\frac{r_i(k_i)}{R_i(r_i(k_i))} > 1 \tag{5.3}$$

holds, duration of activity *i* was percental more decreased than cost increased in iteration k_i. In this case, we denote it “economic reasonable” to crash activity *i* in k_i – although, in practice, it might be sometimes reasonable to crash *i* even if the above ratio is lower than 1 (e.g. for the release of important resources necessary for other activities in another project). However, at a process level, it is more difficult to evaluate the economic efficiency of a particular crashing strategy and to compare it with other crashing strategies. Let us demonstrate this problem assuming the following trivial scenario.

A process comprises five activities of equal cost and duration, all activities are sequentially linked and no feedback relationships exist. In this case, t_{tot}=50 (Figure 5.15a). If a project manager decides to crash all activities by 20%, t_{tot} is shortened by 20% (Figure 5.15b) as well but the individual activity costs rise according to some crashing function. To keep the

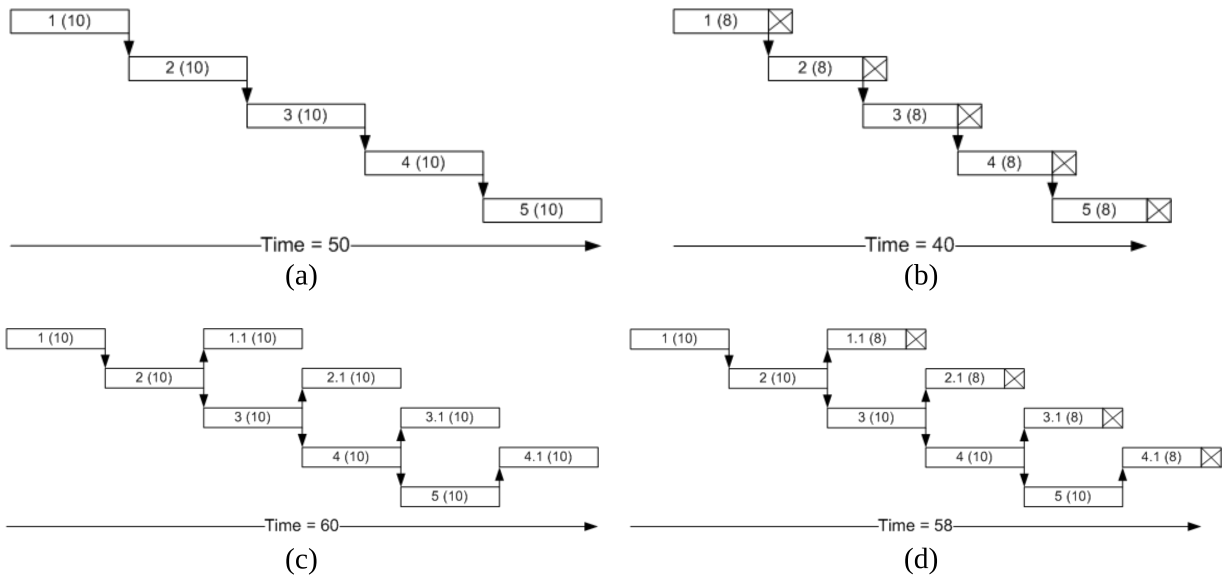


Figure 5.15: Illustration of different processes with activity durations in brackets. a) A simple process without crashing b) a simple process with crashing c) a simple process with iteration and no crashing d) a simple process with iteration and crashing.

example simple, let us suppose a linear relationship in cost rise due to crashing, i.e. any crashed activity costs 20% more than the original activity. Accordingly, process schedule is expedited by 20%, too, and crashing seems to be a "fair" strategy for managers as schedule could be accelerated at a more or less fair price. However, if the process features feedback relationships, which might provoke rework, we face the problem of deciding whether to crash an iterated activity or not. Basically, we could distinguish two extreme crashing strategies with respect to the iterated activities:

Strategy 1: Reworked activities are not crashed at all.

Strategy 2: Reworked activities are crashed.

Which strategy is more economic? Firstly, we observe that most of the reworked activities would be simply executed in parallel to the non-reworked ones due to the underlying process structure, regardless of any crashing policy (Figure 5.15c). Thus, the elongated schedule of the n_A activities in the process is mainly governed by the rework duration of activity $i_{n_A - 1}$ since it cannot overlap with any non-reworked activity.

Concerning strategy 1, the iterations in the process cause a rise in duration by 20% (we neglect rework impact and learning curve effects for reasons of simplicity in this example), compared to the process without iteration and consequently t_{tot} = 60. This result is interesting in that respect as 80% of the tasks had to be executed again and therefore c_{tot} increases by 80%. Contrary, if we pick the more intuitive crashing strategy 2, overall process time almost remains unchanged because of the sequential process structure (Figure 5.15d). As t_{tot} = 58 holds in this case, we note only a moderate schedule acceleration of 3.3% but simultaneously a boost in cost of 8.9%, both compared to strategy 1. Hence, we regard strategy 2 as economic less reasonable than strategy 1. By the way, the best policy for our example would be to crash only the last reworked activity.

Measuring the economic efficiency of crashing

Even this simple example illustrated that the decision to crash an activity or not in a cyclic process environment may not be as obvious and intuitive as for acyclic processes. In order to quantitatively compare different crashing strategies in some of the later test scenarios we introduce the *crashing effect parameter ϕ*:

Definition 5.1 *The crashing effect parameter ϕ defines the ratio between the percental process time change (decrease or increase) compared to original duration due to crashing, and the simultaneous cost increase in percent of original cost:*

$$\phi = \frac{\Delta t_{t_{tot/c}}}{\Delta c_{c_{tot/c}}} \cdot \tau \tag{5.4}$$

where

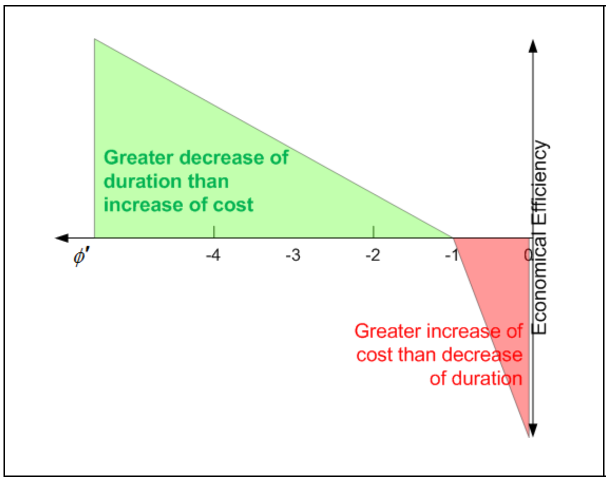
$$\Delta t_{t_{tot/c}} = \frac{t_{tot/c} - t_{tot}}{t_{tot}} \cdot 100 \tag{5.5}$$

holds and $t_{tot/c}$ corresponds to the duration of a process applying the selected crashing strategy. Analogical,

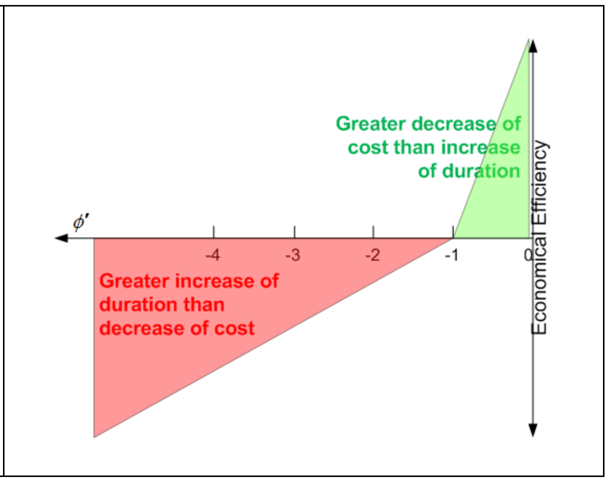
$$\Delta c_{c_{tot/c}} = \frac{c_{tot/c} - c_{tot}}{c_{tot}} \cdot 100 \tag{5.6}$$

denotes the percental change of process cost due to crashing, compared to process cost without crashing. The factor τ represents a weight indicating the priority of ϕ in favor of cost or duration. Finally, we assume that $c_{tot}>0$ as well as $t_{tot}>0$ and define $\phi=-\infty$ or $\phi=\infty$ (depending on $\Delta t_{t_{tot/c}}$) if $\Delta c_{c_{tot/c}} = 0$.

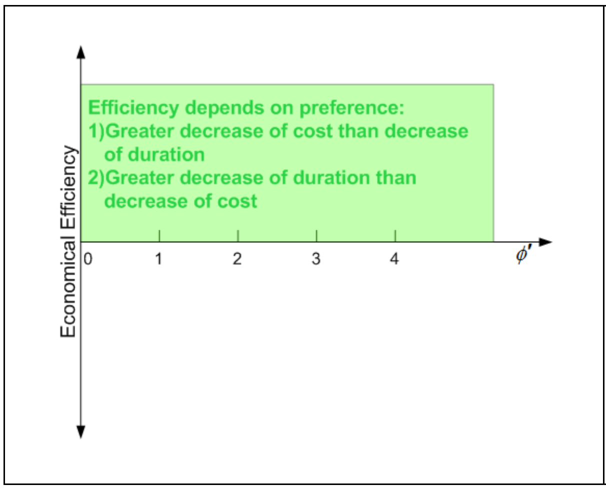
Unfortunately, the validity of ϕ is ambiguous depending on the algebraic signs for $\Delta t_{t_{tot/c}}$ and $\Delta c_{c_{tot/c}}$. Figure 5.16 displays the four cases for different signs (assuming τ=1) while we consider the first case (Figure 5.16a) as the most likely to appear in practice.


Greater decrease of duration than increase of cost
Greater increase of cost than decrease of duration
Economical Efficiency
φ'
-4
-3
-2
-1
0

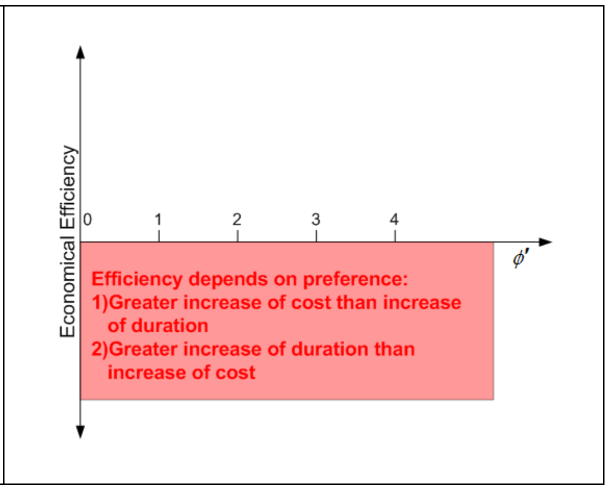
(a) $\Delta t_{t_{tot/c}} < 0$ and $\Delta c_{c_{tot/c}} > 0$


Greater decrease of cost than increase of duration
Greater increase of duration than decrease of cost
Economical Efficiency
φ'
-4
-3
-2
-1
0

(b) $\Delta t_{t_{tot/c}} > 0$ and $\Delta c_{c_{tot/c}} < 0$


Efficiency depends on preference:
1)Greater decrease of cost than decrease of duration
2)Greater decrease of duration than decrease of cost
Economical Efficiency
φ'
0
1
2
3
4

(c) $\Delta t_{t_{tot/c}} < 0$ and $\Delta c_{c_{tot/c}} < 0$


Efficiency depends on preference:
1)Greater increase of cost than increase of duration
2)Greater increase of duration than increase of cost
Economical Efficiency
φ'
0
1
2
3
4

(d) $\Delta t_{t_{tot/c}} > 0$ and $\Delta c_{c_{tot/c}} > 0$

Figure 5.16: Illustration of the ratio $\boldsymbol{\phi}$.

5.2.2 Test setup

To examine the behavior of crashing, we conducted a series of tests using the identical artificial test processes, including the same cost and duration of the 20 activities (Table 5.2), proposed in the previous section. Instead of five work policies, we considered only those three work policies which permit crashing: P_3, P_4, and P_5. Table 5.3 lists the crashing related parameters of our model and their valuation for the next tests. Most importantly, we selected two extreme crashing strategies for the test banks: 1) all activities are crashed in all their iterations by a constant intensity $r_i(k_i)$=0.2 (which is a quite conservative choice) or 2) all activities are not crashed in all their iterations, i.e. $r_i(k_i)$=0 holds for all activities and all iterations. We had to cut down the practically infinite number of possible crashing strategies to those two extreme scenarios in order draw some general conclusions on the behavior of crashing by means of simulation. In the next chapter we will present an optimization approach capable of detecting Pareto-optimal crashing strategies.

In terms of crashing function we decided to apply the more general continuous crashing function (equation 4.18) and simulated different assignments for its factor α. Notably, we set α=13.644 for some of the tests. We obtain this value by solving equation 4.8 assuming $r_i(k_i)$=0.2 as well as $R_i\left(r_i\left(k_i\right)\right)=0.2$. Thus, crashing an activity *i* by 20% will result in both, a time reduction of *i* by 20% and a cost rise of *i* by 20% - which could be called “fair”. Furthermore, we set the Brooks factor to κ_i=1 as it is possible, but unlikely, that crashing elongated the activity.

	Test Parameter			Parameter Value
		Constant	Variable	
Crashing	Crashing intensity - $r_i(k_i)$	X		Either $r_i(k_i)$=0.2 or $r_i(k_i)$=0
	Crashing function - $R(r_i(k_i))$	X		Continuous crashing function
	Factor for $R(r_i(k_i)) - \alpha$		X	α=16, α=13.644, α=8, α=4, α=2
	Brooks factor - κ_i	X		Tests with κ_i=1

Table 5.3: Crashing related model parameters for the sensitivity analysis.

A further crucial remark for the following tests refers to our prohibition of applying crashing more than once to an activity, i.e. we do not allow a potentiation of time reduction for an activity *i* by $r_i(k_i)$ in subsequent iterations. We feel this is a reasonable constraint in order to avoid unrealistic time reductions. Besides, in practice it is unlikely to assign further resources to an activity, which has been already expedited through new resources before, just because it is reworked.

5.2.3 Economic efficiency of crashing

5.2.3.1 Base case

At the beginning of this section we study the impact of process architecture on the behavior of crashing in cyclic process environments. For this purpose, we begin with a base case scenario which is supposed to explore how basic architecture related model parameters, e.g. the number of feedbacks, affect the economic efficiency of crashing. In addition to the described test setup in the precedent subsection, we assumed the following parameters to be constant for the base case:

- Learning and rework impact for reworked activities: λ_{ij}=0.5
- Applied work policy: P_3 without overlapping
- Factor α for the crashing function: α=13.644
- Brooks factor: κ_i=1
- Factor τ=1
- Maximal number of iterations: $\hat{k}_i = 5$

Given these test parameters, $\phi = -1$ would always hold for acyclic processes, i.e. the crashing of the processes would cause a time reduction of 20% and a cost increase by 20%. Though, adding feedbacks in the process changes everything. Table C.1 through Table C.9 in the appendix record the simulation outcomes with respect to ϕ for our base case tests. Thereby,

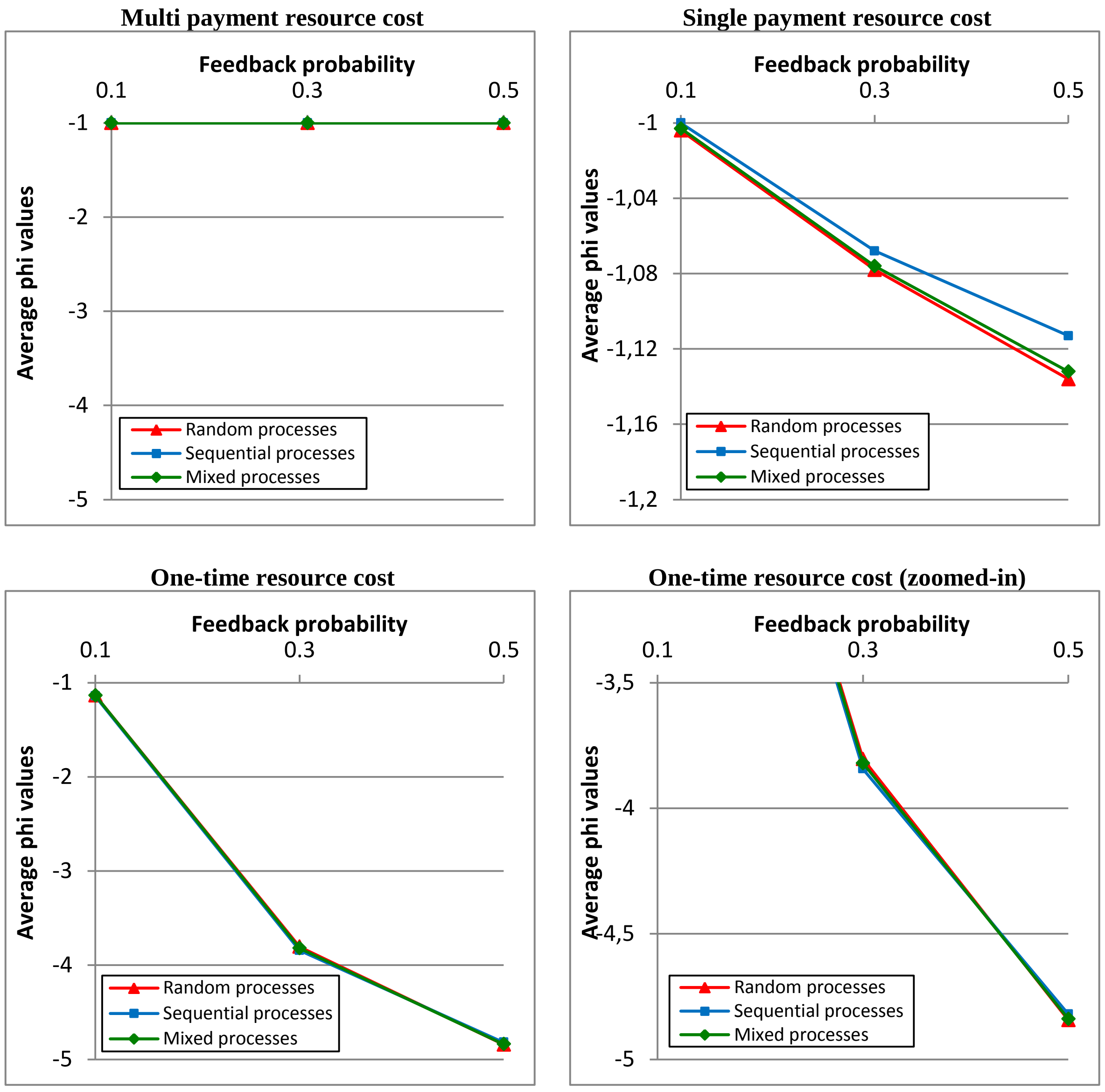


Figure 5.17: Economic efficiency of crashing for the base case.

we simulated test processes with different process architectures including the 16 different combinations for number of feedbacks & feed-forwards, feedback probability and process type. Averaging the 16 ϕ values for a specific feedback probability allows us to measure the effect of feedback on the economic efficiency of crashing. The lower this sum, the more economic is crashing. Figure 5.17 plots these average values for the three distinct payment methods of crashing (see section 4.4) and allows us to conclude the following insights:

1. The economic efficiency of crashing goes up with growing number of reworks (i.e. higher feedback probability; see Figure 5.18) if the resources required for crashing are paid via one-time payment or single-payment. This is not a big surprise as the additional

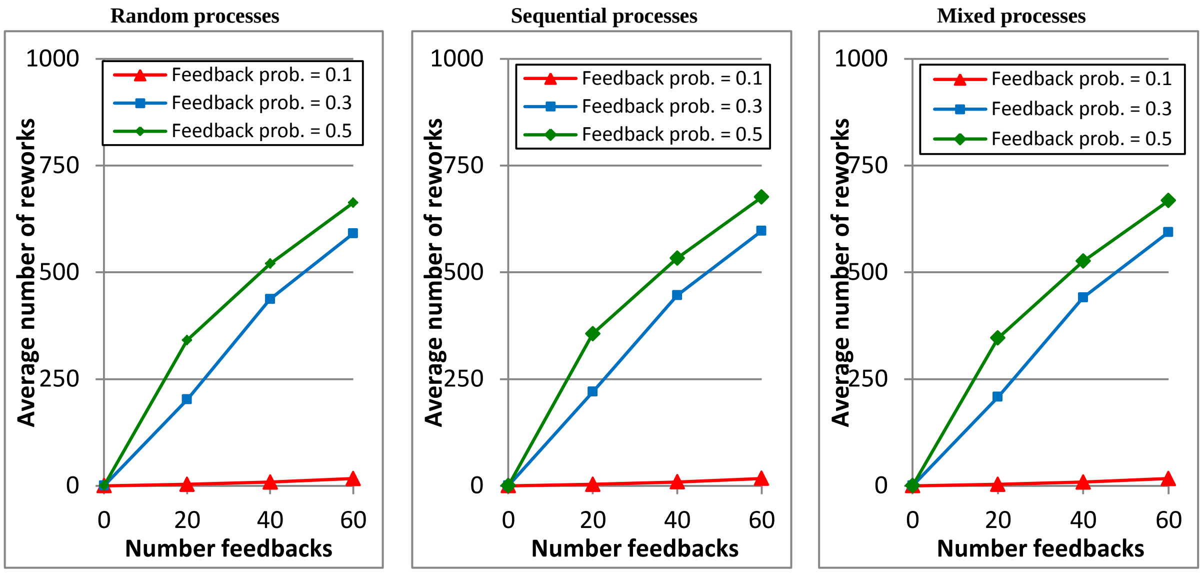


Figure 5.18: Average number of reworks for different processes crashed via multiple payment assuming 120 feed-forwards.

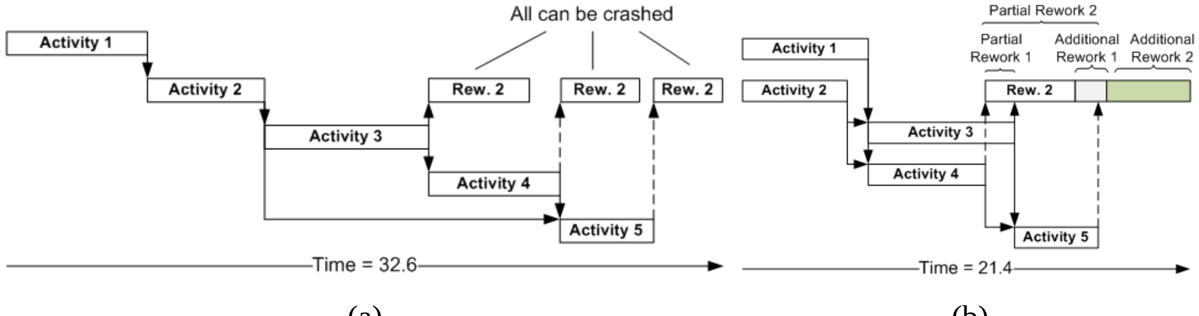
resources must not be paid over the entire time span of an activity. Contrary, $\phi = -1$ for all tests if crashing resources must be paid continuously. As a general observation, we recognize huge discrepancies between the payment modes for crashing. This indicates that the resource type for crashing is a crucial driver for its efficiency.

2. As opposed to acyclic processes, we discovered that process structure (sequential, mixed, random) of cyclic processes affects ϕ as long as crashing resources must not be paid continuously in every iteration (e.g. staff). In case of single payment, we found that crashing becomes more favorable with increasing parallelism within the processes, i.e. random test processes performed in average best and sequential ones worst. This outcome is mainly caused by the highest values for cost increase due to crashing (i.e. $\Delta c_{c_{tot}/c}$) for sequential processes (e.g. Table C.10-Table C.12 in the appendix). At the same time, the percental time reduction through crashing (i.e. $\Delta t_{t_{tot}/c}$) was not affected by any process type. Therefore, sequential processes featured the worst ϕ values. We identified the relatively high difference of reworked activities n_R[19] between sequential

[19] We would like to point out that n_R accounts only for activities which are completely reworked and not for the activities which are partially reworked.

processes and the other two process types as main reason for the higher $\Delta c_{c_{tot}/c}$ outcomes. Assuming an identical amount of feedback relationships, identical activity costs and durations as well as identical feedback probabilities, the higher n_R value for sequential processes is provoked by its lower *activity-density* (i.e. the number of activities executed within a certain period of time) compared to the other process types. Thus, in case an activity *i* produces new input information for another activity *j*, it is more likely that *j* is not executed simultaneously and therefore has to be completely reworked (and crashed) again. Contrary, for highly parallel processes it is more likely that *j* is executed in parallel hence only resulting in a partial rework of *j*, which is cheaper than completely reworking and crashing *j* again. Our thesis is empirically confirmed by Table C.16 through Table C.21 which record the number of reworked activities and the number of partial reworks for all process types assuming single payment crashing in all iterations. Figure 5.19 illustrates this issue with the help of a simple sequential (Figure 5.19a) and mixed (Figure 5.19b) process. In this example, the sequential process structure produces three reworks while the mixed process structure provokes only one rework and two partial reworks.

As for the case of one-time payment crashing, it is clear that absolute crashing costs must be identical for any process structure. Consequently, the outcome of $\Delta c_{c_{tot}/c}$ - and ultimately of ϕ as well since $\Delta t_{t_{tot}/c}$ does not change – is mainly governed by



(a) (b)

Figure 5.19: Comparison of activity density for sequential (a) and parallel processes (b).

absolute process cost c_{tot}: the smaller c_{tot} the higher the impact of crashing cost and thus the higher (=worse) $\Delta c_{c_{tot}/c}$ values. Accordingly, sequential processes appeared to be the most favorable process structure for crashing if feedback probability is low or medium high as c_{tot} is higher compared to random or mixed processes. The $\Delta c_{c_{tot}/c}$ recorded in Table C.13-Table C.15 confirm this statement. In contrast, if feedback probability is high, c_{tot} raises more for processes comprising a high number of parallel activities consequently.

5.2.3.2 Investigating the effects of bottleneck activities

One of our base case assumptions likely to fail in practice constitutes the almost identical values for cost and duration with $c_i / t_i \in [9,11]$ (see Table 5.2). Simultaneously, we noticed due to preliminary tests, which are not presented in this work, that the position of differently loaded activities (regarding its cost and duration) within a process affects ϕ. To account for effects due to different workloads at different positions in the process, we set up test scenarios for the placement of “bottleneck” activities with particularly high cost and duration values in the interval $c_i / t_i \in [19, 21]$. Thereby, we distinguished 1) *front-loaded processes* with five bottleneck activities (i=1,..,5) at the beginning of the process 2) *medium-loaded processes* with five bottleneck activities (i=8,..,12) in the middle of the process and 3) *back-loaded* processes with five bottleneck activities (i=16,..,20) at the end of the process. Furthermore, we fixed feedback probability to 0.5 for all test cases since we do not expect any insights for different feedback probabilities. Relative to the base case, other model parameters remained unchanged. The entire set of simulation results with respect to ϕ is available in the appendix (Table C.37-Table C.45) and revealed the following insights:

1. The distribution of bottleneck activities distinctly affects economics of crashing. Clearly, medium-loaded processes yielded the lowest (=best) average ϕ values and are

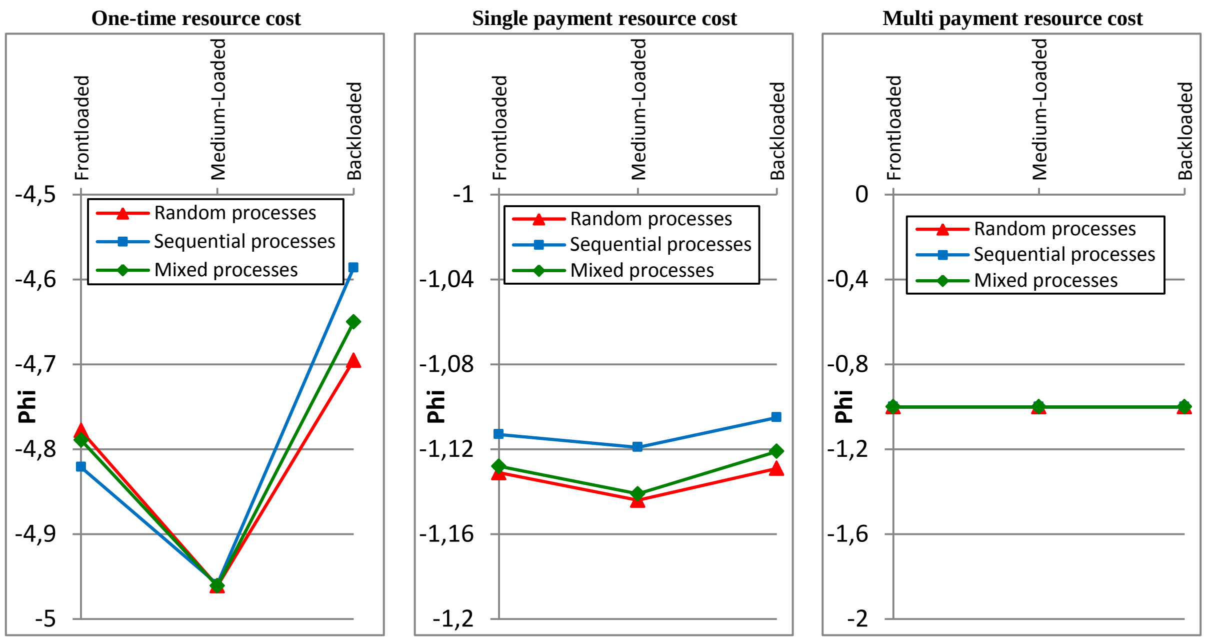


Figure 5.20: Economic efficiency of crashing for processes with bottleneck activities.

most favorable for crashing. Contrary, back-loaded processes do not favor crashing (Figure 5.20). This outcome can be explained with the distribution of reworks, which is exemplary visualized by Figure 5.21 for front-loaded processes with 120 feed-forwards and 60 feedbacks (the equivalent distributions for medium-loaded, back-loaded and equally loaded processes are depicted at Table C.46-Table C.48 in the appendix). Apparently, activities in the middle of the process are statistically reworked more often

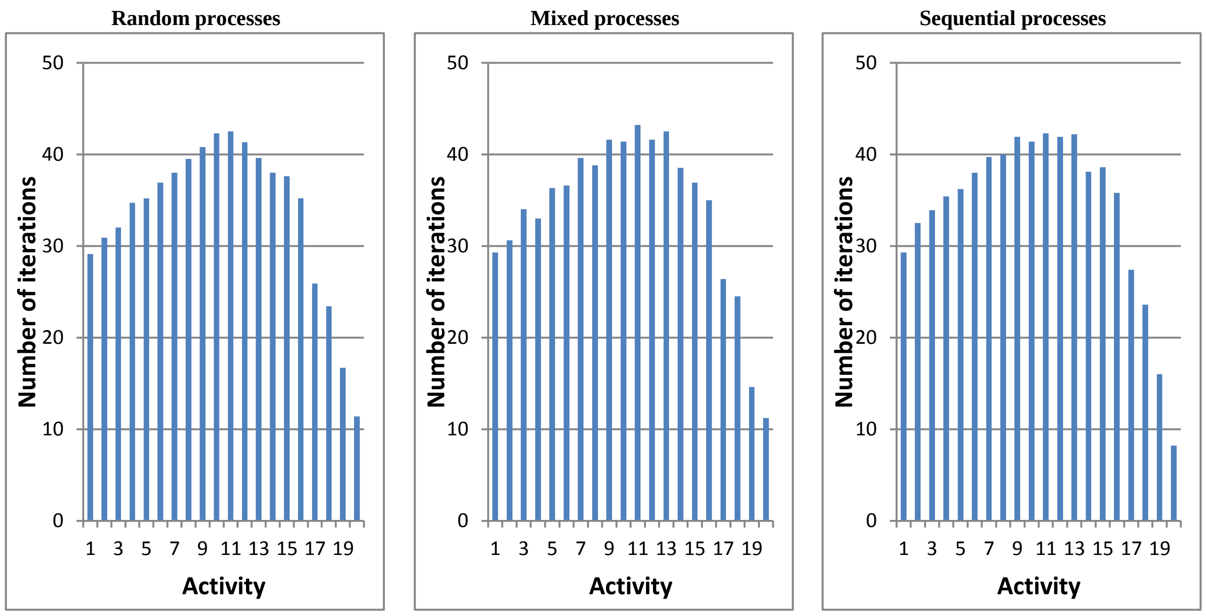


Figure 5.21: Distribution of reworks for front-loaded processes with 120 feed-forwards and 60 feedbacks.

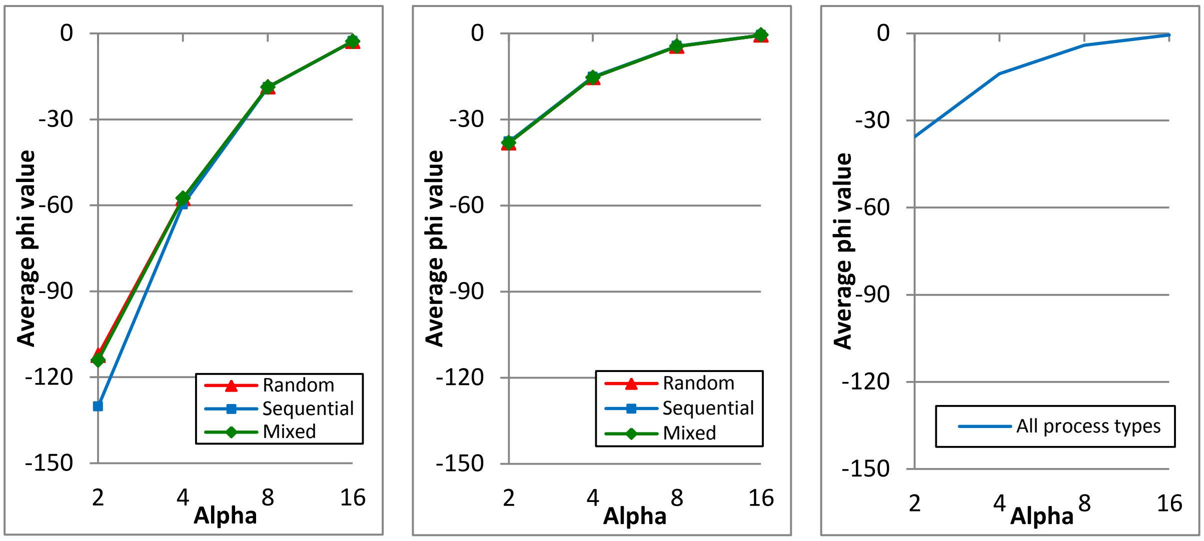
than those at the beginning or at the end of the process thus affecting overall process cost and duration most – regardless of process type or the placement of bottleneck activities. Hence, crashing applied to costly and long-lasting activities in the middle of the process will result in a percental higher process cost/duration reduction compared to crashing of back-loaded or front-loaded processes.

2. Consistent with the base case, crashing is most favorable for processes with high number of parallel activities (i.e. in our test cases random process) and not favorable for sequential process structures, regardless of the placement of bottleneck activities.

5.2.3.3 Investigating the effects of learning and crashing function

Based on the previous time-cost study of process architecture related parameters and work policy we know that λ_{ij} mainly acts as a scaling factor for process cost & duration. Thereby, process cost and duration obviously grow with increasing λ_{ij} (or vice versa decline with decreasing λ_{ij}) as rework is more expensive/time-consuming. Thus, in case of higher λ_{ij} values, the benefits of additional resources are more advantageous (in particular if associated cost arise only in the first iteration of an activity) as they can be used for a longer period of time. Generally, ϕ becomes better (i.e. declines) with increasing λ_{ij} values, and vice versa ϕ increases with declining λ_{ij} values (see Table C.28-Table C.33 in the appendix).

Since crashing cost heavily depend on the defined crashing function, we were also interested in the effects of varying α values for the continuous crashing function (equation 4.8). For this purpose we simulated several scenarios based on the base case parameters. As exception to the base case, we assumed a fixed feedback probability of 0.5 for all processes and tested four different assignments for α: 2, 4, 8 and 16. In view of our selected crashing function it is not surprising that crashing becomes less economic with growing α as illustrated by Figure 5.22 (see also Table C.22-Table C.27 in the appendix). Evidently, a higher α value exponentially rises the price for crashing and, in turn, results in an exponential decrease of ϕ.



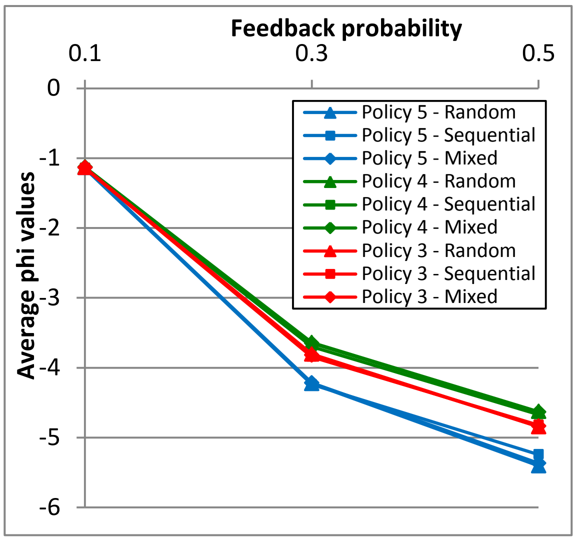
(a) One-time resource cost. (b) Single payment resource cost. (c) Multi payment resource cost.

Figure 5.22: Comparison of average economic efficiency for different alpha values of the crashing function.

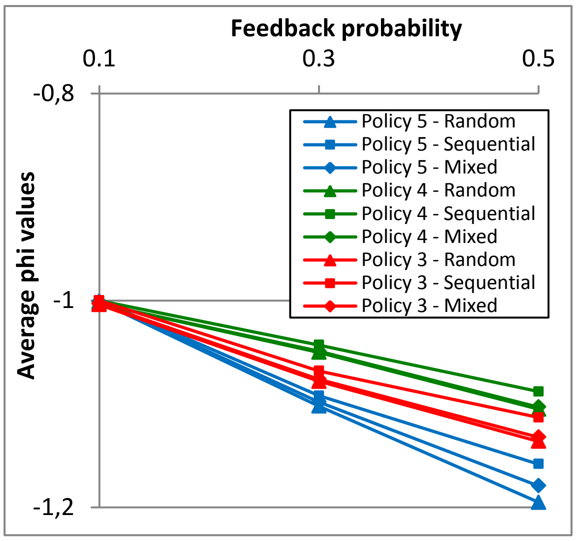
5.2.3.4 *Investigating the effects of work policy*

According to Table 4.1, work policies P_4 and P_5 allow, beside P_3, the application of crashing, too. To investigate whether the rules of these three different work policies may actually change ϕ we performed base case tests for P_4 and P_5 in the same manner as for P_3 and compared the outcomes. As single exception to the base case, we did not consider the multiple payment of resources since $\phi = -1$ is supposed to hold in this case for every test process regardless of work policy. In fact, a visualization of the corresponding test results (Figure 5.23) clearly indicates a correlation between the choice of work policy and ϕ. For both payment methods, P_5 performed best while P_4 proved to be least economic choice.

We claim that P_5 is superior to P_3 and P_4 due to its higher scale-up behavior for cost (see Figure 5.4). In light of identical crashing cost for the two single payment methods, $\Delta c_{c_{tot}/c}$ must decrease with higher process cost according to equation 5.8. In turn, the reduction of $\Delta c_{c_{tot}/c}$ causes a decline of ϕ due to constant $\Delta t_{t_{tot}/c}$ values for all test cases. We argued similar in section 5.2.3.1, explaining the varying ϕ outputs for different process types.



(a) One-time resource cost.



(b) Single payment resource cost.

Figure 5.23: Comparison of base case tests for different work policies.

5.2.4 Summary

The insights gained in the previous sections justified our investigation of crashing assuming cyclic processes. In fact, our analysis confirmed the gaps of past research on crashing mentioned in the first chapter as well as its role for the time-cost tradeoff problem. Our findings can be summarized as follows:

1. As opposed to acyclic processes, the type of resources used for crashing is crucial. In dynamic process environment, managers should generally use resources which have to be paid only once for all potential iterations of an activity, e.g. machinery or software. Then, crashing becomes more economic with increasing number of activities to be reworked in the process. Contrary, adding resources to an activity which must be paid in every of its iterations (e.g. staff) causes a constant economic efficiency regardless of process architecture related process parameters like the number of feedbacks. Though, notably, we did not obtain any test results which would be characterized as inefficient, even if resources had to be paid in any iteration.
2. The primary type of information flow within a cyclic process affects economic efficiency of crashing. Generally, crashing becomes more economic with increasing

number of parallel activities, i.e. activities which are uncoupled with respect to feed-forward information flow. Contrary, processes featuring a strictly sequential information flow performed worst.

3. Crashing should be applied to those activities which 1) exhibit highest cost and duration values and which concurrently 2) exhibit a high probability to be repeated. Tests with artificial processes demonstrated that, in average, activities are most likely to be repeated if they are placed in the middle of a process.

4. In contrast to acyclic processes, work policy impacts the efficiency of crashing as it significantly governs the scale-up behavior for process cost and schedule. Thereby, crashing is favored by work policy rules which provoke high cost. Hence, policy P_5 appeared to be the most appropriate work policy and P_4 the most inadequate one.

5.3 Time-Cost Investigation of Overlapping

5.3.1 Introduction

In principle, overlapping and crashing affect process cost and schedule in an identical manner: both strategies are supposed to shorten lead-time but augment process cost as downside effect. Nevertheless, substantial differences exist:

1. Overlapping can only be applied to sequential or coupled activities.[20] Hence, we expect that the process architecture plays a more pivotal role for overlapping than for crashing.
2. Overlapping requires the consideration of at least two activities: one or more upstream activities and one or more downstream activities. Importantly, the potential expenses for overlapping (i.e. cost for rework) merely affect the downstream activities while cost and duration of upstream activities remain unchanged. In contrast, crashing merely involves an individual activity.
3. The amount of additional rework for the downstream activity, provoked by overlapping, depends on absolute overlapping duration instead on a percental decrease of a single activity's duration/cost like in the event of crashing.

Due to these apparent discrepancies, we decided to separately investigate crashing and overlapping. Yet, the outline for the overlapping related sensitivity analysis in this section is almost identical to that for crashing. Firstly, we provide an overview of the test setup, followed by a presentation of the test results. The section is concluded with a summary of the insights.

Measuring the economic efficiency of overlapping

Prior to the discussion of test results we introduce, analogical to crashing, an overlapping effect parameter ψ to measure the efficiency of overlapping:

[20] If two or more activities are independent of each other, they may overlap as well. However, we do not refer to this case.

Definition 5.2 *The* <u>*overlapping effect parameter* ψ</u> *defines the ratio between the percental change of process time, compared to original duration, due to any overlapping strategy and cost:*

$$\psi = \frac{\Delta t_{t_{tot}/o}}{\Delta c_{c_{tot}/o}} \cdot \varsigma \tag{5.7}$$

where

$$\Delta t_{t_{tot}/o} = \frac{t_{tot/o} - t_{tot}}{t_{tot}} \cdot 100 \tag{5.8}$$

holds and $t_{tot/o}$ *corresponds to the duration of a process applying the selected overlapping strategy. Likewise,*

$$\Delta c_{c_{tot}/o} = \frac{c_{tot/o} - c_{tot}}{c_{tot}} \cdot 100 \tag{5.9}$$

denotes the change of process cost due to overlapping, compared to the process cost without overlapping. Similar to crashing we use a factor ς *representing a weight which indicates the priority of* ψ *in favor of cost or duration. Finally, we assume that* $c_{tot}>0$ *as well as* $t_{tot}>0$ *and define* $\psi = -\infty$ *or* $\psi = \infty$ *(depending on* $\Delta t_{t_{tot}/o}$*) if* $\Delta c_{c_{tot}/o} = 0$.

5.3.2 Test setup

In order to make the test results for overlapping somewhat comparable to those for crashing we considered the same artificial processes (with identical cost & duration for all activities etc.) presented in section 5.1.2. Additionally, we examined work policies P_3-P_5 for the upcoming tests. Though, vitally, we did not permit the use of crashing for P_3-P_5 in order to obtain results which are completely associated to the impact of overlapping and not distorted by any crashing effects.

Table 5.4 lists the overlapping related parameters of our model along with the values assigned to them. To limit the number of test cases, we had to keep those model parameters constant which we expected to be least crucial for the derivation of new insights. Part of these restrictions was also the application of only two extreme overlapping strategies:

1. Overlapping of sequentially coupled activities does not occur at all.
2. Two or more activities which are sequentially coupled overlap – if possible – in any iteration. According to equations 4.5 and 4.6, overlapping cannot occur if an activity would have to deliver its output to an active activity.

In the event of overlapping, we selected the linear overlapping function $h''_{ij}\left(t'_{O_{ij}(k_i,k_j)}\right)$ to calculate rework cost and duration. Due to its simplicity, the linear overlapping function is more suited to demonstrate the effects of overlapping assuming artificial test processes. Finally, we set the various overlapping related points in time to a constant value (see Table 5.4) which allows a comparison with the crashing results to some extent. For instance, we decided to deliver an upstream activity's information to all possible successors after 80% of its completion (similar to crashing the activity by 20%) while a downstream activity receives

	Test Parameter			Parameter Value
		Constant	Variable	
Overlapping	Factor for overlapping function - α		X	α =1, α =0.5, α =0.25, α =0
	Factor for cumulative rework – β	X		β=0.5
	Overlapping function $h_{ij}\left(t'_{O_{ij}(k_i,k_j)}\right)$	X		Linear overlapping function $h''_{ij}\left(t'_{O_{ij}(k_i,k_j)}\right)$
	DSM (minimal information available without penalty) - M_3	X		Constant $m_{3,ij}$ = 1.0
	DSM (maximal information needed without penalty) - M_4	X		Constant $m_{4,ij}$ = 0
	DSM (minimal information available with penalty) - M_5	X		Constant $m_{5,ij}$ = 0.8
	DSM (maximal information needed with penalty) - M_6	X		Constant $m_{6,ij}$ = 0.2
	Time in point when information is delivered - T_D	X		Constant T_D = 0.8
	Time in point when information is received – T_R	X		Constant T_R = 0.2

Table 5.4: Test parameters for work policy P_{3b}.

upstream information just after 20% of its execution.

5.3.3 Economic efficiency of overlapping

5.3.3.1 Base case

Like section 5.2 we start our investigation of overlapping with a base case which basically covers the impact of architecture related model parameters on the efficiency of overlapping. For this purpose, the following assumptions hold:

- Learning and rework impact for reworked activities: λ_{ij}=0.5
- Maximal number of iterations: $\hat{k}_i = 5$
- Applied work policy: P_3 without crashing
- Factor ς=1
- Factor α_{ij}=0.5 for the overlapping function. Assuming the overlapping with T_D=0.8 and T_R=0.2 of two activities, which feature identical cost/duration values, then combined cost of both activities raise by 10% while duration is accelerated by 10% - hence constituting a nearly "fair" overlapping strategy for the two-activity case. Besides, for our sequential test process without feedbacks, overall process duration is expedited by 20% while cost rises by 20% thus making the test results somewhat comparable to crashing (at least for that process type). Though, different to crashing, it is much more difficult to define a fair overlapping strategy on process level as the number of overlapping events, which determine the change of process cost/duration caused by overlapping, is difficult to predict in advance.

Considering these base case rules we simulated the artificial test processes for all 16 distinct n_{ff}/n_{fb} combinations and yielded ψ values recorded in Table D.64 through Table D.66 in the appendix. After analyzing these outcomes, we propose the following insights:

1. In average, applying the base case overlapping strategy to iterative processes becomes more economic with increasing number of reworks (and thus process dynamics). Figure

5.24 highlights how the averaged ψ values for all n_{ff}/n_{fb} combinations decline (i.e. overlapping becomes more economic) for any process type since $\Delta t_{t_{tot}/o}$ decreases more than $\Delta c_{c_{tot}/o}$ goes up with rising feedback probability (Figure 5.25 and Figure 5.26). Remarkably, and contrary to the corresponding crashing tests, $\Delta t_{t_{tot}/o}$ does not remain constant while $\Delta c_{c_{tot}/o}$ increases.

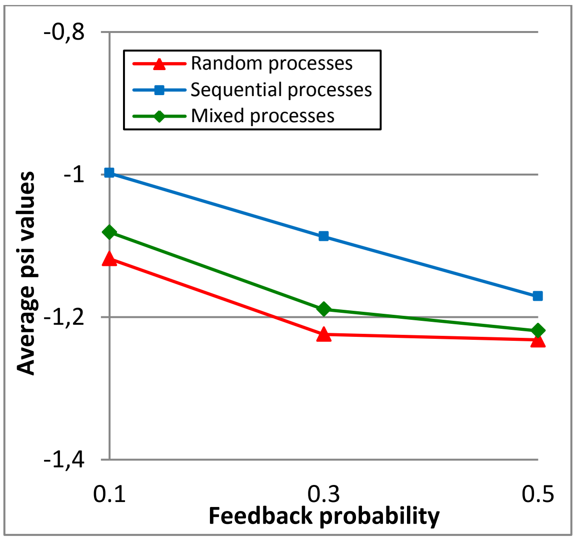


Figure 5.24: Average economic efficiency of overlapping for the base case of policy P_{3b}.

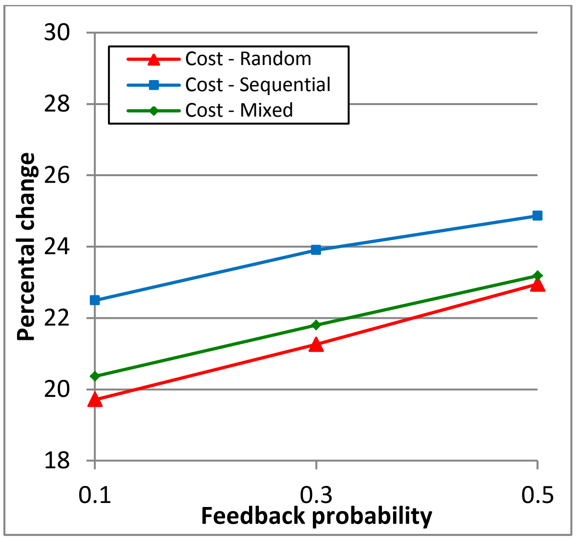


Figure 5.25: Percental change of process cost due to overlapping for the base case of policy P_{3b}.

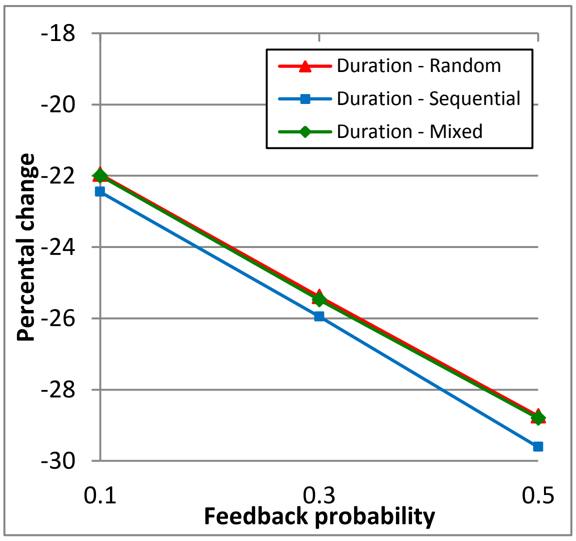


Figure 5.26: Percental change of process time due to overlapping for the base case of policy P_{3b}.

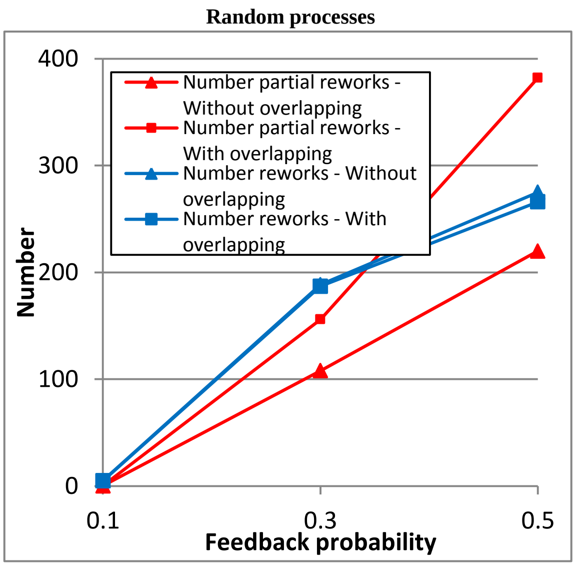


Figure 5.27: Average number of reworks and partial reworks with and without overlapping.

Increased activity density

In order to explain these observations, it is crucial to initially point out that overlapping clearly augmented the number of partial reworks for any of the test processes (Figure 5.27-Figure 5.29) because of an augmented activity density – a phenomenon already noticed throughout our investigation of crashing. Due to overlapping, activities deliver their output earlier to successors while the activities' individual durations are not reduced (in fact, they are even elongated through the additional rework caused by overlapping). Hence, it is more likely that an activity delivers its information to a successor which is still executed than being finished. Figure 5.30 visualizes this issue using a simple process consisting of four sequentially linked activities and three feedbacks. As a result, overlapping "squeezes" the process with less slack times between activities finally causing an exponential boost of partial reworks. On the other hand, the number of "normal" reworks, which are completely reworked (i.e. k_i of an activity i changes), is only marginally affected by overlapping and exhibits a slower (i.e. logarithmical) scale-up behavior.

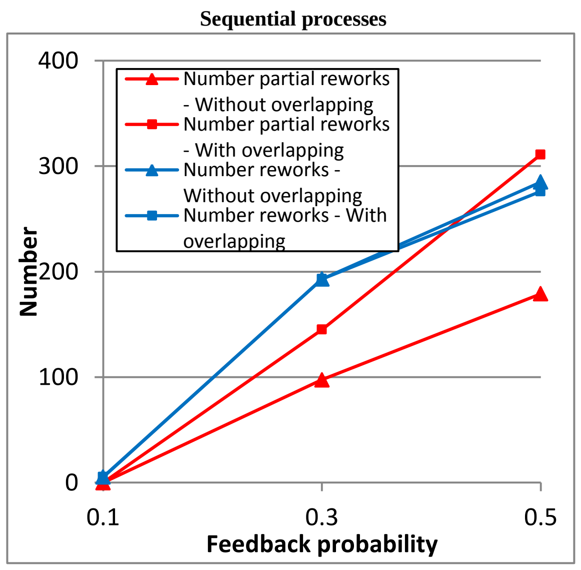


Figure 5.28: Average number of reworks and partial reworks with and without overlapping.

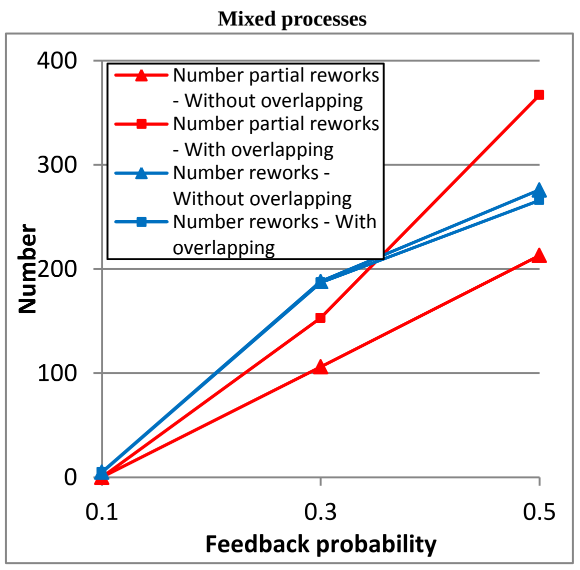


Figure 5.29: Average number of reworks and partial reworks with and without overlapping.

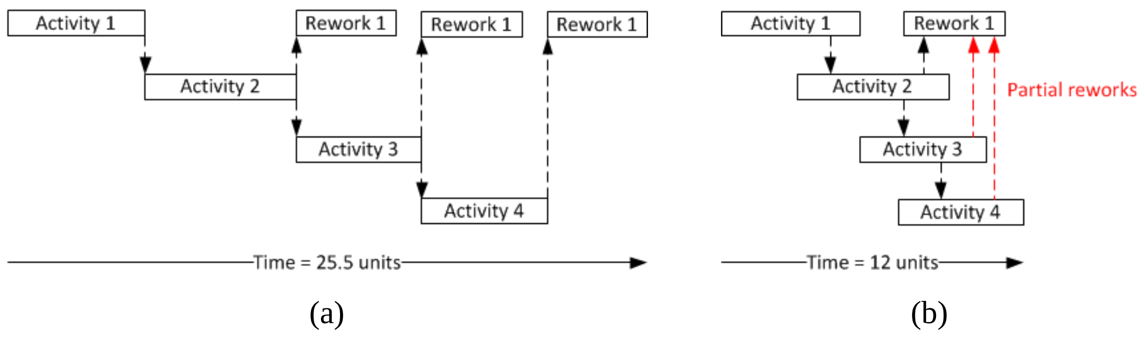


Figure 5.30: Comparison of different activity densities without (a) and with (b) overlapping.

Increase of cost

With the impact of overlapping on partial reworks described, we now proceed to explain the evolution of $\Delta c_{c_{tot}/o}$ and $\Delta t_{t_{tot}/o}$. Given the assumptions of our base case, then $\Delta c_{c_{tot}/o}$ is basically composed of two values: 1) rework cost arising through the overlapping events themselves[21] and 2) cost attributed to partial rework events. Both values rise – and as a consequence $\Delta c_{c_{tot}/o}$ as well – with higher feedback probability (see Figure 5.27-Figure 5.29). As the exponentially increasing partial rework costs are countervailed by the mere logarithmical increase of rework cost associated with actual overlapping events, we note that $\Delta c_{c_{tot}/o}$ grows linearly (Figure 5.25).

Decrease of duration

Keeping in mind the results with respect to ψ (Figure 5.24), $\Delta t_{t_{tot}/o}$ must decrease at a higher rate than $\Delta c_{c_{tot}/o}$ grows. Probably, it is initially not intuitive that $\Delta t_{t_{tot}/o}$ actually declines as the exponential boost of partial reworks for any test process (Figure 5.27-Figure 5.29) is expected to elongate the individual activity durations. Though, the critical path of a process, which determines its overall lead-time, is not necessarily a summation of activity times. In fact, we found that the temporal benefit of overlapping

[21] i.e. $\sum_{i=1}^{n_A} \sum_{j=0}^{\hat{k}_i} \Delta c_{O_i(k_j)}$

two individual activities is even exponentiated (contrary to crashing) while the number of consecutively overlapped activities increases. As the critical path always constitutes a chain of consecutively connected activities[22] this effect holds for the critical path, and thus $\Delta t_{t_{tot}/o}$, too. To visualize our proposition, we extended the simple two-activity overlapping model by two more activities and calculated the relevant overlapping parameters subject to the base case overlapping assumptions (Figure 5.31) as well as identical durations for all activities (10 units).

We claim that, relative to process time, an earlier start of activity *i* due to overlapping with its predecessor *i*-1 effects the entire chain of succeeding activities until the end of the critical path. Importantly, a successor *i*+1 does not commence earlier relative to *i* but relative to process time (except *i* and *i*+1 additionally overlap). Thus, in the example depicted at Figure 5.31, overlapping shortens overall process duration note only by 10% but even by 16.05%.[23] Increasing the number of consecutively linked activities amplifies this effect and makes overlapping even more useful with respect to process duration as demonstrated by Figure 5.32. It depicts the simulated start times for

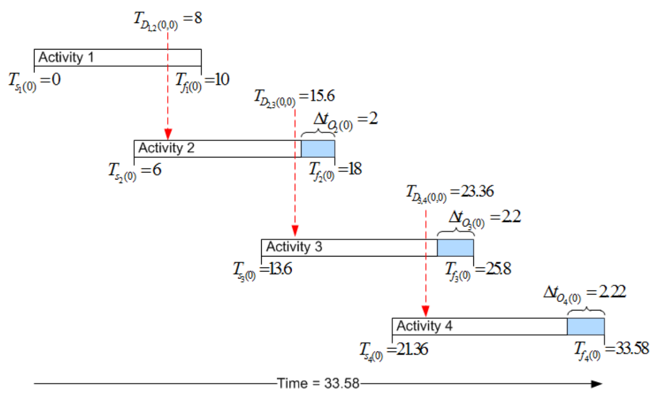


Figure 5.31: Illustration of an overlapped waterfall process consisting of 4 equal activities assuming the base case parameters.

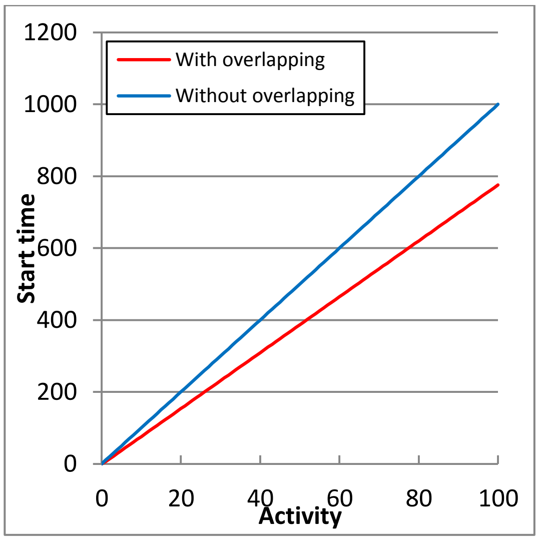


Figure 5.32: Calculation of start times with and without overlapping for a waterfall process consisting of 100 equal activities.

[22] In the simplest case, the critical path merely consists of one single activity.
[23] Without overlapping, overall process duration constitutes 40 units.

100 sequentially linked activities (a “waterfall process”) of equal duration (10 units) without the application of the base case overlapping parameters (red line) and with overlapping (blue line). Clearly, the gap between start times diverges in favor of applying overlapping with increasing number of activities.

2. Similar to crashing, we discovered that process structure affects the efficiency of overlapping. Given low feedback probabilities, overlapping appears to be most economic for random processes and least economic for sequential ones. In fact, sequential processes feature the highest number of overlapping events (=number of reworks; see Figure 5.28), which is the reason for their lower $\Delta t_{t_{tot}/o}$ values compared to mixed or random processes. But this temporal advantage is not for free. Particularly for low feedback probabilities, $\Delta c_{c_{tot}/o}$ is mainly governed by rework cost generated through overlapping events and not by the number of partial reworks. Therefore, sequential processes are more pricy than the other two process types with less overlaps. Differently, however, partial reworks grow exponentially with increasing feedback probability and consequently affect $\Delta c_{c_{tot}/o}$ much more for high feedback probabilities. Since sequential processes exhibited the least increase of partial reworks, the difference between $\Delta c_{c_{tot}/o}$ values of sequential process and $\Delta c_{c_{tot}/o}$ values of random or mixed processes will decline with higher feedback probability. Therefore, ψ values for sequential processes approximate the respective values for random and mixed processes with higher feedback probability.

5.3.3.2 *Investigating the effects of bottleneck activities*

Next, we present the insights related to the placement of bottleneck activities. For this purpose, we did not change the base case overlapping parameters except for the extension of cost & duration for selected activities at the beginning, in the middle, and at the end of the

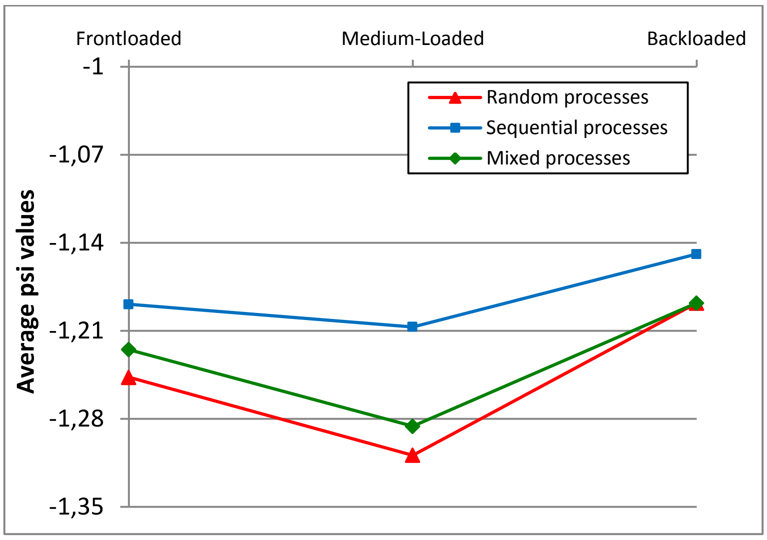


Figure 5.33: Average economic efficiency of overlapping for differently loaded processes.

process. The corresponding simulation outcomes regarding ψ (Table D.91-Table D.93 in the appendix) are similar to those for crashing and can be summarized as follows:

1. Economic efficiency of overlapping is affected by the distribution of bottleneck activities. Again, we gained the best results for medium-loaded processes and the worst ones for back-loaded processes (Figure 5.33). Since activities in the middle of a process are more likely to be repeated than those at the end, the benefits of overlapping become more salient if the highly iterated activities are those which exhibit the highest amount of cost & duration.

2. Consistent with the base case, overlapping is most favorable for processes with high number of parallel activities (i.e. in our test cases random process) and not favorable for sequential process structures.

5.3.3.3 *Investigating the effects of learning and overlapping function*

Regarding the effects of learning and overlapping function, we conducted tests with varying λ_{ij} values as well as different α values for the overlapping function and studied the resulting impact on ψ. Still, the base case assignments hold for all model parameters – except for these

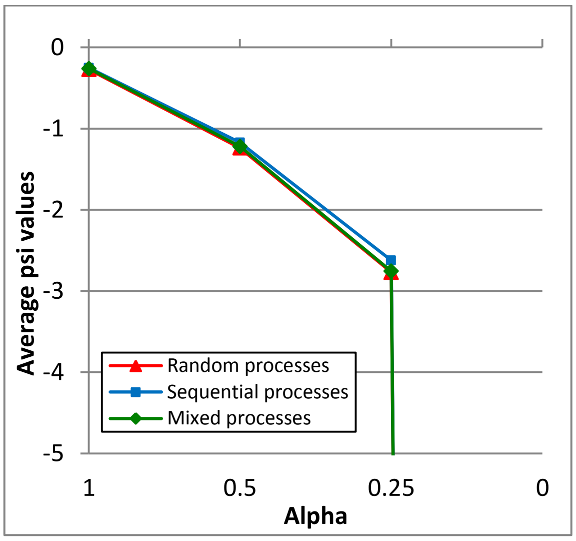


Figure 5.34: Average economic efficiency of overlapping for different alpha values.

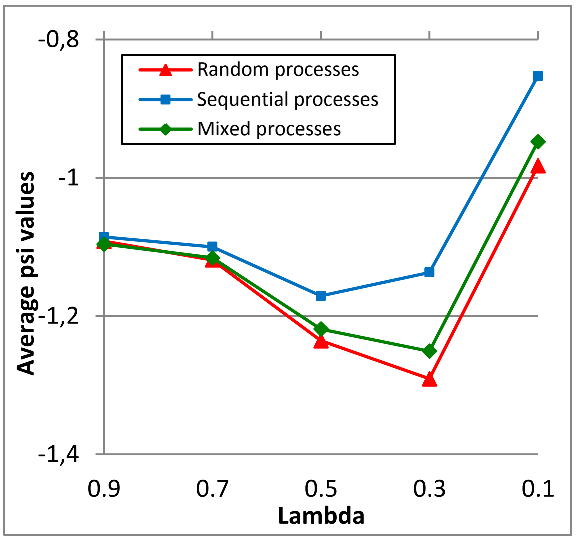


Figure 5.35: Average economic efficiency of overlapping for different λ_{ij} values.

two. Table D.85 through Table D.90 in the appendix list the ψ outcomes for all test processes, and produced the following findings:

1. While it is no surprise that overlapping becomes more economic with decreasing α for the linear overlapping function (as the amount of rework due to overlapping decreases), it is interesting to note that $\psi > 0$ holds even in case of α_{ij}=1 (Figure 5.34). Recalling the definition of the applied overlapping function (equation 4.12), α_{ij}=1 implies an additional rework duration/cost for a downstream activity j equal to the overlapping duration with an upstream activity i, i.e. $\Delta t_{O_j(k_j)} = t'_{O_{ij}(k_i,k_j)}$ holds. In this case, we would intuitively expect that $\Delta t_{t_{tot}/o} = 0$, and consequently ψ=0, holds.

   Though, as a result of overlapping, the downstream activity j may be finished earlier than the upstream activity i, i.e. $t_{f_j(k_j)} < t_{f_i(k_i)}$, assuming we do not consider any rework penalties. In this situation, rework time equal to the overlapping duration could be added to j and still, overall duration would be reduced through overlapping thus generating $\Delta t_{t_{tot}/o}$ values below 0. Figure 5.36 displays this scenario which is likely to

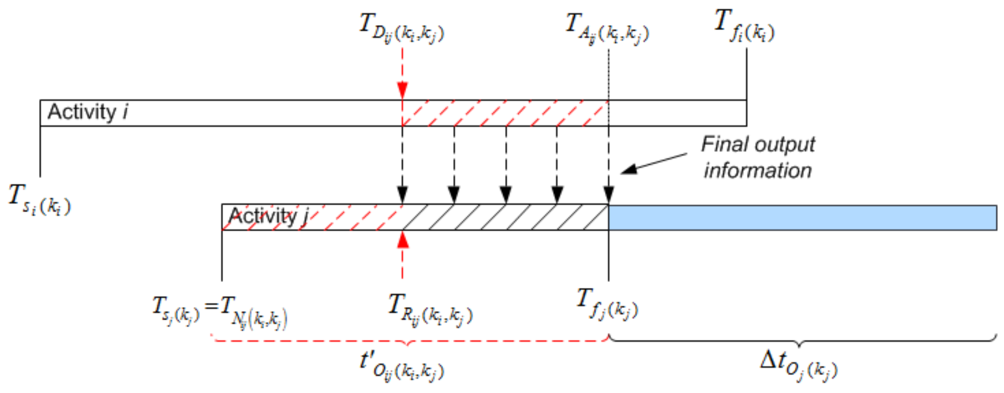


Figure 5.36: Overlapping between two activities assuming an earlier finish of activity *j* than activity *i*.

appear if the duration related difference between two overlapped activities is high (e.g. between a non-reworked activity and a reworked activity).

2. Opposed to crashing, we did not yield a monotonically increasing function for ψ with increasing λ_{ij} values (Figure 5.35). Rather, overlapping becomes for all process types more economic with decreasing λ_{ij} values until a certain inflection point for $\lambda_{ij} \in [0.5, 0.3]$. Then, ψ dramatically rises indicating that our base case overlapping strategy is not economical anymore. Mainly decisive for this behavior are the $\Delta t_{t_{tot}/o}$ values for different λ_{ij} which define a bathtub-similar function (Figure 5.38) while differences with respect to $\Delta c_{c_{tot}/o}$ are less striking (Figure 5.37). Considering the fact

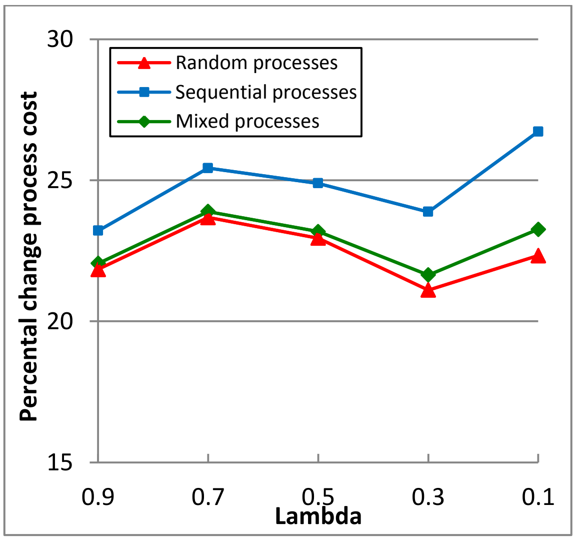


Figure 5.37: Average percental change of process cost for different λ_{ij} values.

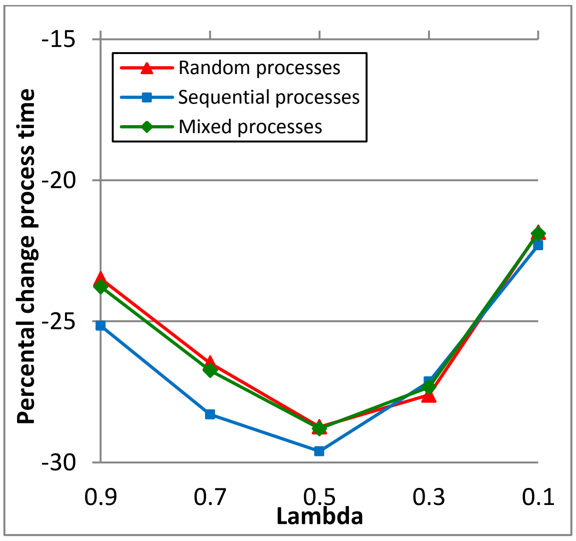


Figure 5.38: Average percental change of process time for different λ_{ij} values.

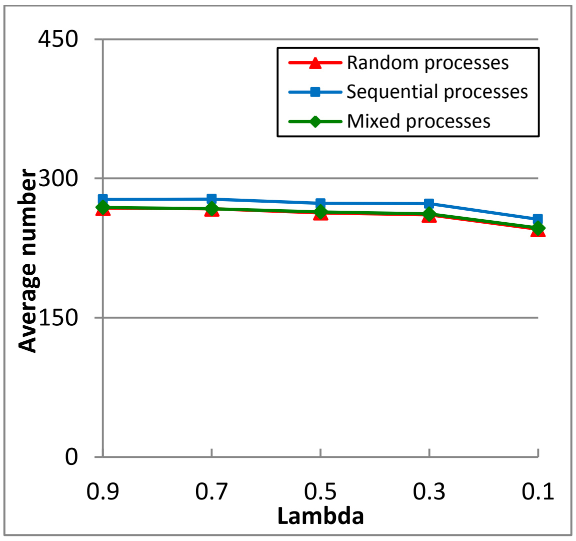


Figure 5.39: Average number of overlapping events for different λ_{ij} values.

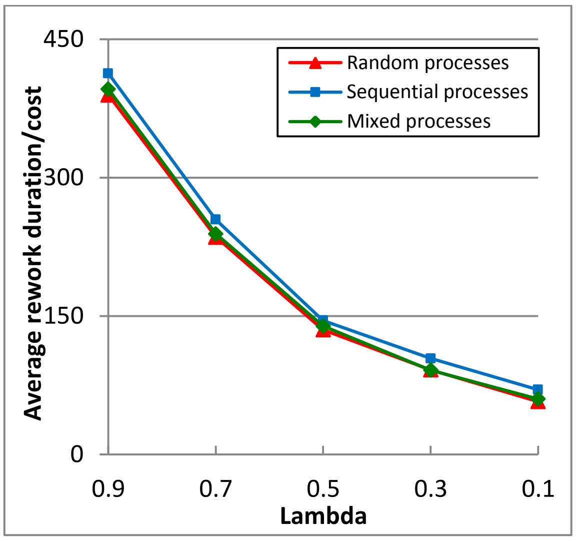


Figure 5.40: Average rework duration/cost due to overlapping events for different λ_{ij} values.

that the average number of overlapping events is almost identical for any λ_{ij} (Figure 5.39), we would intuitively expect that $\Delta t_{t_{tot}/o}$ constantly decreases with increasing λ_{ij}: with higher λ_{ij}, reworked activities retain more of their original duration thus resulting in a higher percentage of rework duration (which can be overlapped) on overall process duration. Though, this effect roughly holds only for $\lambda_{ij} \leq 0.5$.

For λ_{ij}>0.5 we have to take into account that rework duration/cost due to an overlapping of two activities depends on the overlapping duration $t'_{O_{ij}(k_i,k_j)}$ (as pointed out earlier in section 4.3). Due to equations 4.1 and 4.2 (defining the effects of λ_{ij}) $t'_{O_{ij}(k_i,k_j)}$ will decrease exponentially on the basis of λ_{ij} with increasing number of iterations for the activities which overlap. Hence, the average overlapping duration grows exponentially with λ_{ij}, and consequently associated rework duration/cost as well (Figure 5.40). Therefore, $\Delta c_{c_{tot}/o}$ increases with higher λ_{ij} and ultimately depresses the efficiency of overlapping.

5.3.3.4 Investigating the effects of work policy

Assuming the base case parameters for work policy P_{3b} we also simulated the test processes for policies P_4 and P_5 and yielded the ψ values recorded in Table D.106 through Table D.111 in the appendix. The average ψ for all n_{ff}/n_{fb} combinations is visualized at Figure 5.41 clearly proving that work policy has a significant impact on the efficiency of overlapping.

As opposed to crashing, we found that work policy P_4 is most favorable for overlapping while P_5 turned out to be the least economic choice. In case of work policy P_5 it is also interesting to note that ψ does not steadily drop with higher feedback probability but rather increases for higher feedback probabilities than 0.3. We explain these two observations with the fact that P_4 fosters the occurrence of "normal" reworks which can be overlapped while P_5 provokes a higher number of partial reworks which cannot be overlapped. In light of these facts, overlapping can have a greater impact on processes applying P_4 instead of P_5 or P_{3b}.

5.3.4 Summary

Generally, the previously presented test results indicate that overlapping can be regarded as a robust approach to cut process duration, in particular for dynamic processes. Even for adverse conditions, managers can assume that the application of overlapping will positively affect its schedule. Simultaneously, however, it is almost certain that overlapping will unfavorably

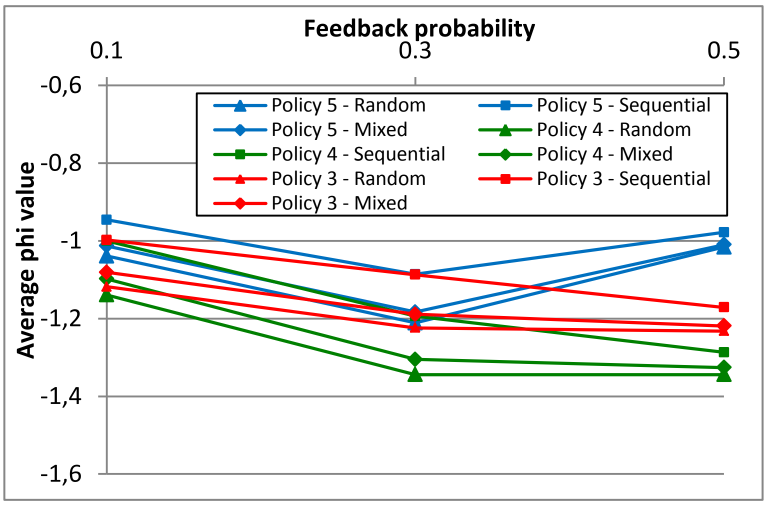


Figure 5.41: Average economic efficiency of overlapping for different work policies.

boost process budget. More in detail, we can conclude the following propositions:

1. With increasing dynamics, overlapping becomes (in most cases) more economic, mainly because of an extended critical path length. We found that a higher number of equally long activities on the critical path amplify the temporal benefits of overlapping two activities. This effect may outweigh the percental cost increase associated with overlapping, which exhibits a linear growth. Interestingly, overlapping slightly reduces the number of reworks, but on the other hand causes an exponential enlargement of partial reworks which make the process significantly more expensive. For work policies fostering the occurrence of partial reworks in order to reduce process time, such as work policy P_5, the cost for partial reworks may even exceed the time consuming benefits of overlapping for certain conditions ultimately diminishing the efficiency of overlapping.
2. The primary type of information flow affects efficiency of overlapping. While sequential processes featured the highest speed-up of process lead-time, they also showed the highest amount of additional cost. Vice versa, overlapping positively affected the schedule of random processes least, but caused the least amount of rework cost. Overall, however, we obtained the best trade-off results between time reduction and boost of cost for random processes, and the worst ones for sequential processes.
3. Contrary to crashing, higher learning and rework impact values must not necessarily enhance the economic efficiency of overlapping. Rather, we gained the best results for an interval $\lambda_{ij} \in [0.5, 0.3]$ where overlapping performed best. Higher or lower λ_{ij} values reduced the economic efficiency.
4. In contrast to acyclic processes, work policy impacts the efficiency of overlapping as it determines the amount of activities which can be overlapped. Thereby, the impact of overlapping is higher for work policy rules which foster the occurrence of “normal” reworks and minimize the number of partial reworks. Therefore, policy P_4 appeared to be the most appropriate work policy and P_5 the most inadequate one.

5.4 Summary

At the beginning of this thesis, we presented the omissions of previous research on the classical time-cost tradeoff problem. This chapter was supposed to close some of these shortcomings, and it actually did. We revealed a series of new, interesting, insights which could moreover serve as a starting point for future research. In the following we list the most important test results in tabular form and summarize the major statements.

First of all, on a macro level, we studied the time-cost behavior of our process model with respect to process architecture and work policies. An overview of the time-cost effects of the tested parameters can be found at Table 5.5 and Table 5.6. As major managerial insight, we demonstrated that time-cost tradeoffs may not only arise due to crashing, overlapping or process architecture but also due to the choice of a work policy if the underlying process

	Parameter	**Effects on cost**	**Effects on time**
Work policy	P_1	Least cost of all work policies due to least amount of rework	Highest duration of all work policies due to least amount of iterative overlapping
	P_2	Second highest cost of all work policies due to second highest amount of iterative overlapping.	Delay of output delivery causes less iterative overlapping, and thus less cost, than P_2 and P_5
	P_4	Delay of output delivery causes less iterative overlapping, and thus less cost than P_2 and P_5	Delay of output delivery causes less iterative overlapping, and thus longer processes than P_2 and P_5
	P_5	Highest cost of all work policies due to highest amount of iterative overlapping and thus rework.	Provokes shortest processes of all work policies due to highest amount of iterative overlapping and thus parallelism.

Table 5.5: Summary of work policy related effects on process time and cost.

	Parameter	**Effects on cost**	**Effects on time**
Process Architecture	M_1 - n_{fb}	Logarithmic increase of cost regardless of work policy: ≈$\boldsymbol{O}(\log(n_{fb}))$	Logarithmic increase of time regardless of work policy: ≈ $\boldsymbol{O}(\log(n_{fb}))$
	M_1 - n_{ff}	Linear increase of cost regardless of work policy assuming $n_{fb}>0$: ≈$\boldsymbol{O}(n_{ff})$	Linear increase of time regardless of work policy assuming $n_{fb}>0$: ≈ $\boldsymbol{O}(n_{ff})$
	$m_{1,ij}$	Higher feedback probabilities increase absolute cost but not scale-up behavior	Higher feedback probabilities increase absolute time but not scale-up behavior
	λ_{ij}	Typically a scaling factor; Depends on n_{fb} and definition of the learning curve	Typically a scaling factor; Depends on n_{fb} and the definition of learning curve
	k_i	Depends on learning curve and n_{fb}; Can cause logarithmic up to an exponential increase of cost: $\boldsymbol{O}(\log n_{fb})$ - $\boldsymbol{O}(2^{n_{fb}})$.	Depends on learning curve and n_{fb}; Can cause logarithmic up to an exponential increase of time: $\boldsymbol{O}(\log n_{fb})$ - $\boldsymbol{O}(2^{n_{fb}})$.
	S	Affects n_{fb} and n_{ff} → see effects for M_1	Affects n_{fb} and n_{ff} → see effects for M_1

Table 5.6: Summary of process architecture related effects on process cost and time.

	Parameter	**Effects on economic efficiency**
Crashing	M_1 - n_{fb},n_{ff}	In average, crashing becomes more economic for all payment types with increasing number of reworks and thus increasing sum $n_{fb}+n_{fb}$.
	M_1 – Process Structure	Process structure affects economic efficiency if crashing resources must not be paid continuously in any iteration. In case of one-time payment for the crashing resources, crashing is most economic for sequential processes if feedback probability is low. For higher feedback probabilities, random processes turned out to be more economic. In case of single payment crashing, crashing is most economic for random processes and least favorable for sequential ones.
	$m_{1,ij}$ – Feedback probability	In average, crashing becomes more economic for all payment types with increasing number of reworks and thus higher feedback probability.
	λ_{ij}	Efficiency of crashing increases with higher λ_{ij} values.
	Bottleneck activities	In average, medium-loaded processes are most suited for crashing while back-loaded processes are least economic.
	Factor for $R(r_i(k_i))$ – α	Efficiency of crashing decreases with higher α values.
	Work policy P_3-P_5	Work policy rules affect the efficiency of crashing. P_5 appeared to be the most favorable work policy for crashing and P_4 the most inadequate one.

Table 5.7: Summary of crashing related effects on process cost & time as well as on its efficiency.

features feedbacks. Likely, this observation was not discovered in the past since prior research mainly assumed acyclic process models. Furthermore, we derived some general time-cost scale-up behaviors for cyclic processes in dependence of various process parameters.

Subsequently, we examined the effects of iteration on crashing and overlapping since such a study has not been conducted yet, too. Firstly, to measure the impact of iterations as well as that of other model parameters on the economic efficiency of crashing & overlapping, we introduced efficiency ratios for both strategies. According to these ratios we claim that, generally, feedbacks in the process amplify the positive effects of both strategies, i.e. crashing and overlapping become more economical with increasing feedbacks. Although exceptions exist, this insight is probably the most important one and it theoretically encourages the application of Concurrent Engineering techniques in dynamic process environments. While the actual impact of feedbacks on crashing and overlapping might be identical, the reasons are different and have been explained in great detail. The results also point out the validity of systems theory for process- and project management: while the effects of crashing and overlapping on time and cost might be predictable on an activity (i.e. system element) level, they are not on a process level (i.e. system level). Thus, in our case it is accurate to say that the system is more than the sum of its individual elements. Table 5.7 and Table 5.8 provide a summary of the effects of considered model parameters on economic efficiency of crashing and overlapping. As a major limitation of our study, we merely considered two extreme crashing/overlapping strategies in order to draw generally accepted conclusions. Beside, by pure means of simulation it is unlikely to determine a best crashing/overlapping strategy. Though, the next chapter of this thesis presents an approach to identify Pareto-optimal crashing and overlapping strategies.

	Parameter	**Effects on economic efficiency**
Overlapping	M_1 - n_{fb},n_{ff}	In average, overlapping becomes more economic with increasing number of reworks and thus increasing sum n_{fb}+n_{fb}.
	M_1 – Process Structure	Process structure affects the efficiency of overlapping. Given low feedback probabilities, overlapping appears to be most economic for random processes and least economic for sequential ones. With higher feedback probabilities, sequential processes become more favorable.
	$m_{1,ij}$ – Feedback probability	In average, overlapping becomes more economic with increasing number of reworks and thus higher feedback probability.
	λ_{ij}	For all process types, efficiency of overlapping initially increases with higher λ_{ij} until a certain inflection point in the interval $\lambda_{ij} \in [0.3,0.5]$ is reached. Then, efficiency decreases with increasing λ_{ij}.
	Bottleneck activities	In average, medium-loaded processes are most suited for overlapping while back-loaded processes are least economic.
	Factor α_{ij}	Efficiency of overlapping decreases with higher α values.
	Work policy P_3-P_5	Work policy rules affect the efficiency of overlapping. P_4 appeared to be the most favorable work policy and P_5 the most inadequate one.

Table 5.8: Summary of overlapping related effects on process cost & time as well as on its efficiency.

Part III

TIME-COST TRADEOFF OPTIMIZATION

6. A New Time-Cost Optimization of PD Processes

To determine Pareto-optimal solutions with respect to time and cost of cyclic PD processes, we do not aggregate these two objectives into a single one (which would reduce a multi-objective problem to a single-objective one). While easy to implement, aggregation methods are incapable of producing certain portions of the Pareto front in a single run. They also require additional problem-specific knowledge about the relative weightings of the various objectives. Furthermore, decision makers often prefer to consider multiple solutions in light of other circumstances and variables which are not part of the formal problem. Therefore, we seek to maintain the PD process time-cost tradeoff problem as a multi-objective optimization problem rather than collapse it into a single-objective optimization problem (which could be done by assuming a monetary value of time, for example). For this purpose, this chapter proposes the components and mechanics of a multi-objective Genetic Algorithm – the ε-MOEA (Deb et al. 2003) – tailored to approximate the time-cost Pareto front of PD processes assuming the novel process model in chapter 4. Finally, we explain how to deal with problem specific constraints.

6.1 Literature Review and Selection of Multi-objective Genetic Algorithm

By surveying the literature, it becomes obvious that most MOGA designs, especially those proposed in the 1980s and 1990s, do not account for both, preservation of diversity and convergence to the true Pareto-front, but merely cope with one criterion. The first implementation of a MOGA dates back to 1984 and is credited to Schaffer (1984) with his thesis on the vector evaluated GA (VEGA). Later, Goldberg (1989) suggested the use of *Pareto ranking* and selection in combination with a niching mechanism to move a population towards the Pareto-front. In this approach, the GA population is decomposed into non-dominated fronts, and individuals within each non-domination level are assigned a common fitness rank. Many publications based on this design appeared in the 1990s. Fonseca and Fleming (1998) proposed the Multi-Objective GA (MOGA), Srinivas and Deb (1994) the Non-dominated Sorting GA (NSGA), Deb et al. (2002) the NSGA-2, and Zitzler et al. (1999, 2002) the Strength Pareto Evolutionary Algorithm (SPEA) and SPEA2, respectively. All of these approaches focus on a good distribution of solutions without ensuring convergence to the Pareto front. On the other side, Rudolph and Agapie (2000) and Hanne (1999) developed MOEAs that guarantee at least some solutions belonging to the global Pareto front. As opposed to the former MOGAs, both algorithms fail in maintaining a good distribution of identified solutions resulting in incomplete coverage of the efficient frontier (Laumanns et al. 2002).

More recent approaches attempt to address both, diversity and convergence. Laumanns et al. 2002 introduced an archiving strategy based on ε-dominance and claimed that this strategy guarantees convergence towards the Pareto front as well as a good solution spread. Based on the idea of applying ε-dominance to MOEAs, Deb et al. (2003) proposed further enhancements to Laumanns' original approach. In a comprehensive, comparative study, Deb et al. (2005) thoroughly tested several MOGAs on artificial test functions with respect to critical performance measures like computational time, hyper-volume, sparsity, and convergence. These test functions, presented in Deb (1999) were constructed to feature multi-objective difficulties including convexity, non-convexity, discontinuity, and non-uniformity. While no strategy dominates throughout all tests, the *ε*-MOEA (Deb et al. 2003) performed best overall, averaged over all test functions and all performance measures. Despite these excellent results for the *ε*-MOEA, knowledge about the Pareto set is still required to some extent. Thus, none of the MOEAs to date can seriously guarantee a rapid and diverse convergence to the true global Pareto front without problem specific knowledge. In fact, choosing a multi-objective GA constitutes a separate multi-objective problem itself. Though, we decided to apply the *ε*-MOEA to the time-cost tradeoff problem due to the encouraging results in (Deb et al. 2005) and its expected ability to cope with noisy fitness functions (like a simulation) in a better way than most other MOEAs because of its use of the *ε*-dominance criterion. In the following, we will present the flow of the *ε*-MOEA as well as its customization to the time-cost tradeoff problem.

6.2 Flow of the *ε*-MOEA

Initialization

Initially, the *ε*-MOEA randomly generates a certain number of chromosomes, each of them representing a unique PD process with different activity sequences or overlapping- and crashing intensities. These chromosomes are afterwards evaluated by a fitness function which assigns them values for process cost and duration using the PD process simulation introduced in chapter 4. Subsequently all chromosomes are split up into two distinct populations. Contrary to the “classical” GA design (see Figure 3.8), the *ε*-MOEA maintains two co-evolving populations during its execution – a parent population, *P*, and an archive population, *A*. While *P* is allowed to contain both, dominated and non-dominated chromosomes with respect to their absolute values for process cost and duration, the archive population is exclusively composed of *ε*-non-dominated chromosomes.

Advantages and disadvantages of *ε*-dominance

Applying the *ε*-dominance (see definition 3.6) criterion to the archive population results in two major benefits (Deb et al. 2005). First, the cardinality of the Pareto-region is reduced as *ε*-dominance decomposes the objective space into multiple hyper-boxes. Spanning a length of ε_i in the i^{th} objective, these hyper-boxes can be occupied by at most one chromosome. Therefore, the number of solutions in the *ε*-Pareto-front goes up with decreasing *ε*. The second benefit of the *ε*-dominance approach is its pragmatism, making the *ε*-MOEA interactive with a decision-maker. Nevertheless, this strategy has disadvantages as well: the *ε* value determining the solution resolution has to be set manually or adaptively. But setting *ε* inappropriately could result in an archive preserving poor solutions. In worst case, the archive would contain only one solution which *ε*-dominates all other solutions.

Selection, crossover and mutation

With the two population sets finally produced, the *ε*-MOEA proceeds as follows. According to a selection method for *P* and *A*, described later in section 5.3, one chromosome is picked out of *P* and one out of *A*. Then, these two chromosomes undergo the ordinary GA mechanisms crossover and mutation, which are explicated in section 5.3, to create a new chromosome (offspring). Divergent to the original flow of the *ε*-MOEA we additionally included two mechanisms after the mutation phase. First, we incorporated an efficient repair mechanism (described in section 5.5) to address firm predecessor constraints in the activity sequence. Second, the uniqueness of the offspring is thereafter controlled in order to avoid duplicates in the two populations as a result of the use of a simulation as fitness function (see section 5.2). In case the offspring is unique it is passed to the fitness function. Otherwise, it is discarded and the algorithm returns to the archive selection procedures.

Acceptance procedure and convergence

As final step in the *ε*-MOEA, acceptance procedures for *P* and *A* (Deb et al. 2005) decide whether the offspring replaces any of its members or not. This procedure is iterated until any predefined convergence criterion is met and the archive population is expected to contain the Pareto-solution in the end. Defining an appropriate convergence criterion has been always a problem in the design of MOGAs in the past (Laumanns et al. 2002, Kumar and Rockett 2002) and no unique solution to it has been proposed yet. Since we do not have any knowledge about the Pareto front in advance, we decided to choose the following convergence criterion. The *ε*-MOEA is assumed to be converged if the absolute number of Pareto solutions remains constant for a certain number of generations and if additionally the number of changes regarding the optimization objectives (i.e. in our case time and cost) within the set of Pareto solutions (e.g. 5% of all Pareto solutions) does not exceed a predefined threshold for this period.

Schematic flow

Figure 6.1 illustrates the overall flow of the selected *ε*-MOEA design. In contrast to most multi-objective GAs, which process the entire population through discrete steps, our version of the *ε*-MOEA constitutes a steady-state evolutionary algorithm (EA). To speed up computation, steady-state EAs use only a portion of the population for processing, and each offspring is compared with the parent population immediately after its creation.

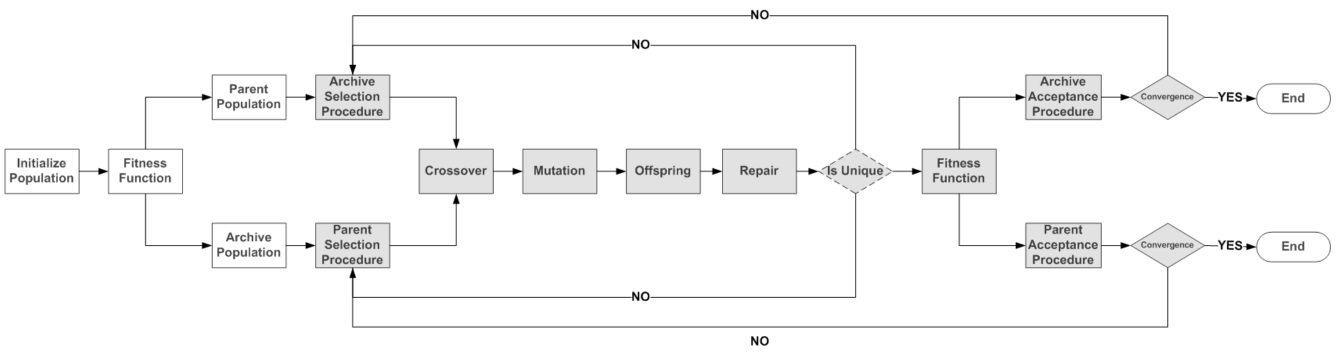


Figure 6.1: Flowchart of the modified ε-MOEA.

6.3 GA components

With the flow of our tailored GA established we now propose the problem-specific setup of its components, i.e. the data structure and fitness function, more in detail. Since the data structure of the GA depends on the data to be optimized, we firstly define the set of parameters for the time-cost optimization.

6.3.1 Selection of optimization parameters

At the beginning of the optimization process, we have to identify those parameters in our PD process model which are considered to be constant and those parameters which are supposed to be optimized. Basically, we decided to encode three fundamental process parameters, namely activity sequence, crashing and overlapping. Notably, we did not include the choice of work policy in our optimization approach as we intended to investigate the impact of work policy on the Pareto-front. Thus, we the following list of model parameters was encoded:

1. The activity sequence $\boldsymbol{S}$
2. The crashing intensity $r_i\left(k_i\right)$ for each activity *i* and every iteration k_i. (up to $\hat{k}_i$)
3. The percentage of activity *i*'s duration when it delivers the output in the k_i-th iteration for activity *j* in the k_j-th iteration: $o_{D_{ij}\left(k_i,k_j\right)}$ in every iteration k_{i} (up to $\hat{k}_i$) for every pair of activities *i* and *j*.
4. The percentage of activity *j*'s duration when it receives the output in the k_i-th iteration from activity *i* in the k_i-th iteration: $o_{R_{ij}\left(k_i,k_j\right)}$ in every iteration k_{i} (up to $\hat{k}_i$) for every pair of activities *i* and *j*.

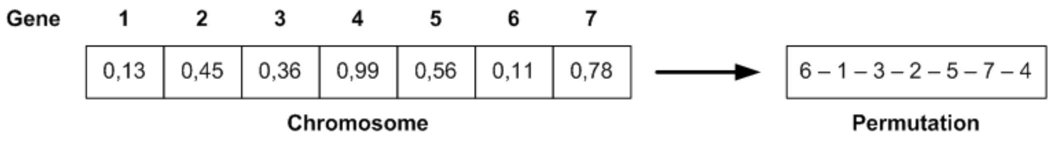
The parameters $o_{D_{ij}\left(k_i,k_j\right)}$ and $o_{R_{ij}\left(k_i,k_j\right)}$ are used to calculate the overlapping duration as well as other overlapping related values, e.g. the resulting rework duration, as explained in section 4.3.

6.3.2 Data structure

Random key encoding

As for the selected optimization parameters, we realize that $r_i(k_i)$, $o_{D_{ij}(k_i,k_j)}$ and $o_{R_{ij}(k_i,k_j)}$ constitute real numbers while ***S*** represents a permutation. In general, permutations are represented through integers in the chromosome. Though, Bean (1994) proposed a random key representation based on real numbers as encoding scheme and pointed out its benefits for combinatorial problems. Mainly for this reason we selected random keys to encode the activity sequence of a PD process.

Typically, the real numbers are used as ascending sorting keys to encode a permutation. These numbers are initially determined by random and alter only under the influence of mutation and crossover. Accordingly, a permutation of length *l* consists of a vector $\boldsymbol{P} = (r_1, r_2, ..., r_l)$ with $\boldsymbol{P} \in [0,1]^l$. Considering the surjective function $\sigma : S_l \to S_l$ with $S_l = \{0,1,...,l-1\}$, random keys are sorted in ascending order: $r_{\sigma(1)} \le r_{\sigma(2)} \le ... \le r_{\sigma(l)}$. A permutation is then encoded by $(\sigma(1), \sigma(2), ..., \sigma(l))$. This formalism describes that the key positions within ***P*** are ordered according to their absolute value. As an example, consider the following chromosome and the corresponding encoding:



Theoretically, a key could be generated several times, although the chance is reduced when using sufficient precision for the keys. In this case, a left-right strategy could be used to ensure feasibility. In addition to the preservation of feasibility, another benefit of random keys is important: partial relative ordering is preserved after crossover. Furthermore, the absolute ordering information is also preserved to some extent (Knjazew 2002).

Encoding of activity sequence

As a result of the random key encoding for ***S***, we treat all optimization parameters as real numbers which can be easily transformed into binary strings with certain degree of precision. At a high level, we divided the binary chromosome into four disjoint segments, each of them representing one single optimization parameter, as follows. First, the process sequence can be easily encoded through n_A random keys representing the identities of the corresponding activities. Each random key is assigned to exactly one distinct gene and consists of 32 bits. Hence, the real number of the key is determined by transforming the binary string to a decimal number which is subsequently divided through 2^{32}. To process only feasible process sequences, we have to take care of two issues: 1) an integer value of the permutation must occur exactly once, and 2) we are not allowed to violate predecessor constraints in ***S***. Fortunately, we can avoid the first problem by the use of random keys. However, dealing with predecessor feasibility will be covered later in section 5.3. For now, we continue explaining the encoding of the two overlapping parameters.

Encoding of overlapping

As the overlapping intensity between two activities *i* and *j* may differ depending on their underlying iteration it is not sufficient to encode $o_{D_{ij}(k_i,k_j)}$ or $o_{R_{ij}(k_i,k_j)}$ through one single bit number. Instead, we have to encode the $o_{D_{ij}(k_i,k_j)}$ or $o_{R_{ij}(k_i,k_j)}$ values for all possible combinations of k_i and k_j. Overall, $(\hat{k}_i+1)\cdot(\hat{k}_j+1)$ combinations exist and must be included in the chromosome. Thereby, each combination is binary encoded through a bit number of any length, n_B, depending on the precision required by the user. We feel that 256 discrete intervals, i.e. an 8-bit encoding (n_B=8), should be sufficient in practice. To reduce computational effort, we furthermore encode only values with $o_{D_{ij}(k_i,k_j)}<1$ and values for

$o_{R_{ij}(k_i,k_j)} > 0$, respectively. Otherwise, we would include unnecessary information merely extending chromosome length.

Thus, the second segment of the chromosome is established by initially identifying those relationships in M_3 and M_5 which meet the aforementioned requirements. Then, we consecutively generate an 8-bit number for every (k_i, k_j) pair as depicted at Figure 6.2 and append it to the existing chromosome. To decode the overlapping segment, every of its n_B-bit blocks is transformed to an integer value u in the interval $\left[0, 2^{n_B}-1\right]$ yielding a value for $o_{D_{ij}(k_i,k_j)}$ in the interval $\left[m_{3,ij} - \frac{u}{2^{n_B}-1}\cdot\left(m_{3,ij}-m_{4,ij}\right), m_{3,ij}\right]$ and, with respect to $o_{R_{ij}(k_i,k_j)}$, in the interval $\left[m_{5,ij} + \frac{u}{2^{n_B}-1}\cdot\left(m_{6,ij}-m_{5,ij}\right), m_{6,ij}\right]$.

Encoding of crashing

The last segment of the chromosome addresses crashing. We encode the crashing value of an activity i in any of its iterations (up to $\hat{k}_i$) using n_B -bit numbers as well. For evaluation, these bit numbers are transformed to an integer value u with $u \in \left[0, 2^{n_B}-1\right]$ in order to generate crashing intensities as follows: $r_i(k_i) = \hat{r}_i \cdot \frac{u}{2^{n_B}-1}$.

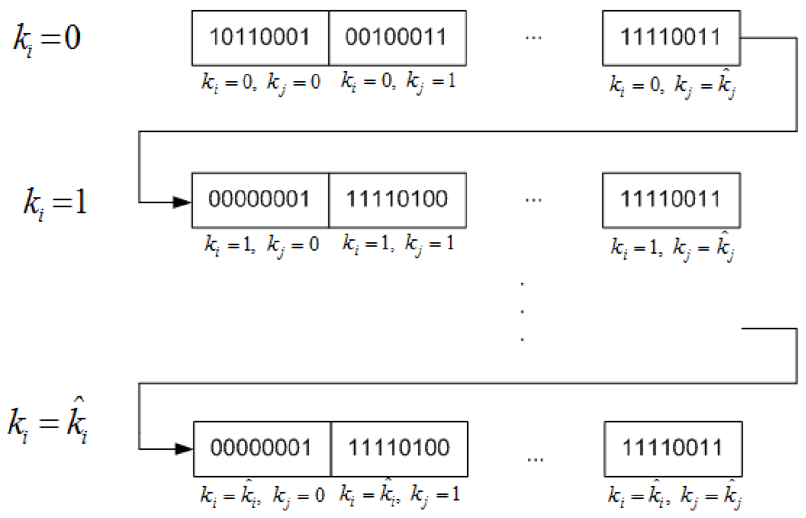


Figure 6.2: Illustration of overlapping encoding.

Aggregating the four different segments of our data structure so far, we obtain an encoding scheme for a chromosome of the multi-objective GA as illustrated at Figure 6.3.

6.3.3 *Fitness function*

Since our process model is too complex to deterministically approximate time and cost of

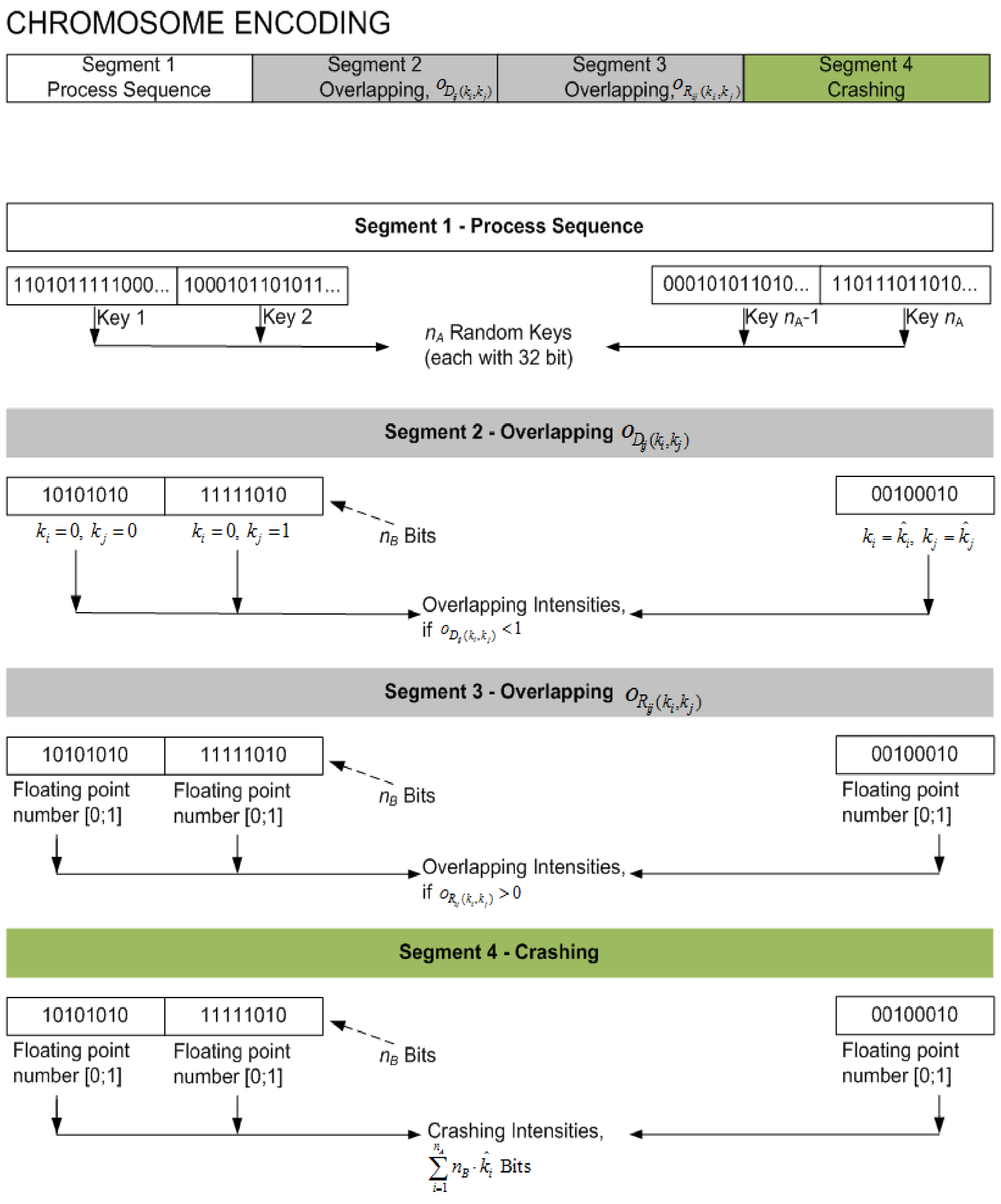

CHROMOSOME ENCODING
Segment 1 Process Sequence
Segment 2 Overlapping, $o_{D_j(k_i,k_j)}$
Segment 3 Overlapping, $o_{R_{ij}(k_i,k_j)}$
Segment 4 Crashing
Segment 1 - Process Sequence
1101011111000...
1000101101011...
000101011010...
110111011010...
Key 1
Key 2
Key n_A-1
Key n_A
n_A Random Keys (each with 32 bit)
Segment 2 - Overlapping $o_{D_j(k_i,k_j)}$
10101010
11111010
00100010
$k_i = 0,\ k_j = 0$
$k_i = 0,\ k_j = 1$
$k_i = \hat{k}_i,\ k_j = \hat{k}_j$
n_B Bits
Overlapping Intensities, if $o_{D_j(k_i,k_j)} < 1$
Segment 3 - Overlapping $o_{R_{ij}(k_i,k_j)}$
Floating point number [0;1]
Overlapping Intensities, if $o_{R_{ij}(k_i,k_j)} > 0$
Segment 4 - Crashing
Crashing Intensities, $\sum_{i=1}^{n_A} n_B \cdot \hat{k}_i$ Bits

Figure 6.3: Illustration of chromosome encoding.

arbitrary PD processes, we had to use the simulation introduced in chapter 4 to gain realistic cost and schedule outcomes.

Drawbacks of a simulation as fitness function

Though, the use of simulation has two major drawbacks. First, the computational effort for simulation becomes greater with increasing number of feedbacks. Thus, a single fitness function evaluation might be extremely time-consuming, which is particularly disadvantageous for a population-based search (like GAs). Second, a simulation can be regarded as a “noisy” fitness function because its output – the resulting c_{tot} and t_{tot} distributions – may slightly vary each time. A noisy fitness function will potentially assign different fitness values to identical chromosomes, leading to varying Pareto-fronts over successive runs of the ε-MOEA. Moreover, identical chromosomes could become competitors in the population of a single ε-MOEA run due to their different fitness values, allowing them both to take up space in the archive population. In worst case, the population could merely consist of multiple instances of the same chromosome with different fitness values. However, two mechanisms prevent this scenario.

Preserving diversity

The first one is part of the MOEA design: the ε-dominance criterion. If cost and duration outcomes for two chromosomes with identical encoding differ by less than the corresponding ε values both chromosomes compete for the same hyper-box. Therefore, one of the two identical chromosomes is discarded, thus ensuring the uniqueness of the other chromosome. Nevertheless, the difference in simulation outcome could also exceed the ε value, particularly if ε is chosen too low; therefore, the stopping criteria for the number of simulation runs must be coordinated with the choice of ε. If this scenario occurs frequently, diversity in the archive, and consequently the number of potential Pareto-solutions, decreases. Fortunately, we can easily avoid this case by extending the ε-MOEA with an operator in front of fitness evaluation (see Figure 6.1) verifying the uniqueness of the offspring by comparing its data structure with all members of *P* and *A*. Obviously, computational time will be increased (since, theoretically, no offspring could be produced for numerous iterations), but we nevertheless emphasize the incorporation of this additional mechanism to ensure diversity.

6.4 GA Mechanics

6.4.1 Selection and acceptance procedures

As pointed out earlier in this chapter, the ε-MOEA maintains two distinct populations instead of one which requires us to apply a selection operator to each of them. Moreover, we have to consider multi-objective fitness values, i.e. chromosomes rather exhibit a separate fitness value for cost and duration than a single aggregated one. Thus, we have to perform two-objective domination checks in order to select a chromosome. In the original study on the ε-MOEA, Deb et al. (2003) proposed the following selection procedures.

Selection

With respect to the parent population *P*, chromosomes are selected according to a *tournament selection* strategy. Tournament selection constitutes a so called ordinal-based selection scheme and is widespread in use due to the ability to ensure an adequate selection pressure over time. Basically, the selection pressure indicates how many copies of the best fit chromosome survive the selection stage (Bäck 1994). During tournament selection, a certain number of chromosomes depending on the size *s* of the tournament are randomly picked. Generally, the best fit chromosome – in our case, the chromosome which dominates the other chromosomes in the tournament – wins the tournament with a certain probability and overcomes the selection phase. If the tournament contains non-dominated chromosomes, then we randomly pick one out of them assuming an equal probability. The literature distinguishes tournament selection with replacement (TWR) and without replacement (TWOR). In the case of TWR all chromosomes which just participated in a tournament are also allowed to be potential candidates for the succeeding tournament. In consequence, the best string is expected to obtain *s* copies in average. Contrary, TWOR requires *s* "iterations" of selection, i.e. individuals are allowed to participate in a tournament only if all other chromosomes have been part of a tournament in the same iteration. Accordingly, the best chromosome gets exactly *s* copies for further processing. Sastry and Goldberg (2001) compared both methods claiming a superiority of TWOR as it is less noisy and needs a lower population size to achieve the same accuracy as TWR. Regarding tournament size *s*, we decided to set $s = 4$ due to the results in (Goldberg et al. 1991, Goldberg and Deb 1991). In contrast to *P*, we just randomly select a chromosome from the archive population *A* without the use of any selection operator.

Acceptance procedures

In addition to the selection procedures, the ε-MOEA features two so called acceptance procedures determining whether the offspring replaces any member in *P* or/and in *A*. Basically, the offspring replaces a chromosome in *P* if it dominates one or more of them (chosen at random) or if it is non-dominated to all population members in *P*. Though, if at least one member in *P* dominates the offspring, it is not accepted. The acceptance procedure for the archive population is more complex and the reader may refer to (Deb et. al 2003) for a detailed description. However, in principle this procedure is similar to the acceptance strategy for *P* but based on ε-dominance checks.

6.4.2 Crossover

In lieu of applying one crossover operator to the entire chromosome, we decided to use crossover for each segment of our data structure separately. This is motivated by the great length of a chromosome – easily comprising several thousand bits – and its time consuming evaluation via simulation. Applying multiple crossover operations leads to a more intense mixing of the genes hence increasing the “search speed” of the ε-MOEA. Though, such a strategy bears the risk of premature convergence. Overall, from a practical point of view, we feel that the computational advantage outweighs the theoretical disadvantage in solution quality.

Knjazew (2002) empirically investigated different crossover operators on artificial test functions claiming a superiority of the single-point crossover (Goldberg 1989) compared to the popular two-point crossover and uniform crossover. Mainly for this reason we also incorporated it in our GA design. In general, the single-point crossover for two chromosomes works as follows. First of all, two chromosomes out of the mating pool (i.e. the population after selection) are randomly chosen and only undergo crossover with a certain crossover

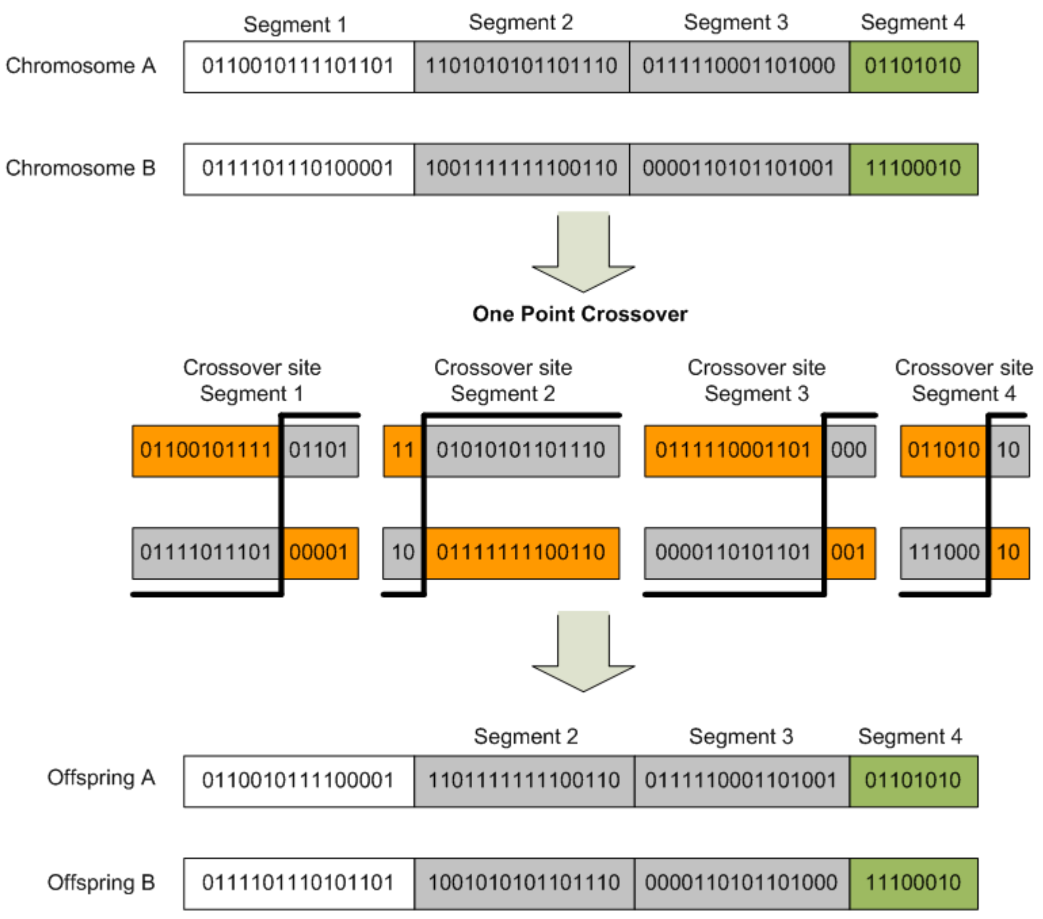


Figure 6.4: Demonstration of crossover.

probability p_c . Then, a crossover sites is chosen randomly. Subsequently both chromosomes are sliced at the crossover site into two segments. Finally, the offspring gets two segments from different chromosomes. Figure 6.4 demonstrates how the crossover works if applied to each of the four sections of our data structure.

6.4.3 Mutation

Similar to crossover, we apply mutation to the four segments of a chromosome separately assuming a certain mutation probability p_m which is typically set to a low value. For this purpose, a simple bit-swap mutation (Goldberg 1989) is supposed to work fine.

6.5 Dealing With Predecessor Feasibility

In order to exclusively process feasible activity sequences after crossover and mutation, we must address predecessor constraints. Firm predecessor constraints are inherently recorded in M_4 and M_6 as an activity j cannot be started before its predecessor i if it requires all input information from i to begin, i.e. if $m_{4,ij} = m_{6,ij} = 0$ assuming $i>j$. As $m_{6,ij}$ constitutes an upper bound for $m_{4,ij}$, $m_{6,ij} = 0$ is sufficient to indicate firm predecessor constraints. Since random key encoding of the activity sequence merely ensures the prevention of duplicate activity IDs in the chromosomes, an additional, efficient repair mechanism is required to transform any activity sequence into a precedent-feasible one. Apparently, a straight-forward repair strategy in case of predecessor violation would be the iteration of crossover or mutation until all constraints are satisfied. Though, depending on the number of constraints, the discrepancy between the immense number of possible permutations (n_A! for n_A activities) and the number of feasible solutions could be great and simply too time consuming. Besides, this strategy is not deterministic. Therefore, we handle predecessor constraints in another deterministic and efficient way as follows.

Strategy to ensure predecessor feasibility

Generally, predecessor conflicts do not occur between activities which can be executed concurrently – i.e., activities which do not rely on predecessor information at the same point in time. Assuming we start with an empty activity sequence at time T_0, we can calculate the set of parallel activities at T_0 based on the DSM M_6 and subsequently pick an activity out of this set according to a deterministic strategy. For instance, we could scan the (potentially infeasible) activity list of the chromosome from left to right and select the activity ID which matches first any activity ID in the set of parallel activities. Then, the chosen activity is appended at the end of the feasible activity list and all of its relationships within the network of activities are temporarily deleted. Repeating this procedure until all activities have been assigned to a spot in the activity sequence, we will never violate any predecessor constraints.

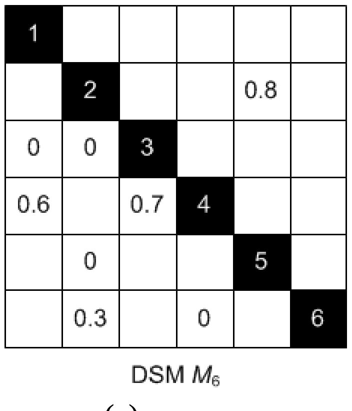
```
Input:

Integer      n_a;
ActivityList p[n_a];
Array        M_6[n_a][n_a];

Output:

Feasible schedule list f[n_a]

Algorithm:

Integer i=0;
Integer j=0;
ActivityList f=new ActivityList[n_a];
WHILE i < n_a
    FOR j to n_a-1
       IF p[j].numberOfPredecessors == 0 AND
          P[j].isScheduled == FALSE THEN
               f[i]=p[j];
               BREAK;
       ENDIF
       j++;
    ENDFOR
    j=0;
    FOR j to M_6.length-1
       M_6[j][f[i].columnID]=0;
    ENDFOR
    j=0;
    p[f[i]].isScheduled = TRUE;
    i++;
ENDWHILE
```

Figure 6.5: Pseudo-code for mapping any permutation to a feasible activity list.

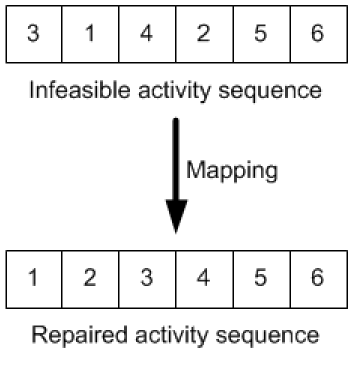
The pseudo-code shown in Figure 6.5 describes the repair mechanism more in detail. As input, the algorithm needs the permutation to be mapped, `p`, the number of activities, `n_a` and the DSM `M_6[n_a][n_a]`. Besides, two auxiliary variables `i` and `j` are used. The output is a precedent- feasible activity list `f`. The algorithm identifies the first activity of `p` without precedent activities and assigns it to spot `i` in `f`. Then, all dependencies on the selected activity are deleted in `M_6[n_a][n_a]`. This simple algorithm scales up in complexity $\boldsymbol{O}(n_A^2)$.

Exemplary repair of predecessor violation

As an example, consider the DSM in Figure 6.6a representing M_6 and an infeasible activity sequence to the right of it. The DSM indicates the precedence relationships between the activities – e.g., activity 1 must precede activity 3 and activity 2 must precede activities 3 and 5. Accordingly, the permutation {3-1-4-2-5-6} is not feasible since, for instance, activity 3 is scheduled prior to activity 2. Applying the repair algorithm leads to the following results. Activities 1 and 2 do not depend on any other activities in the set and thus comprise the initial set of parallel activities. The first value in this set which also occurs in the infeasible sequence `p` is {1}. Thus, the first value of the feasible schedule list, `f`, must be 1: `f`[0] = 1. After deleting the row and column for activity 1 in M_6, the next iteration of the algorithm begins, detecting a new set of parallel activities: {2}. In this set, activity 2 is the earliest one in `p` and



(a)



(b)

Figure 6.6: Preserving predecessor feasibility.

consequently `f[1] = 2` holds. The row and column for activity 2 are deleted and a new loop starts. Repeating all steps of the algorithm until convergence, we obtain the precedent-feasible schedule list `f = {1, 2, 3, 4, 5, 6}`.

7. Case Studies

The following chapter presents the combined application of the previously proposed new model, simulation and optimization strategy on two real-world PD processes: the first process appeared in the aerospace industry and the other in the automotive industry. Thereby, the goals of this study are twofold. Firstly, as much as possible insights gained during the sensitivity analysis, which were based on artificially constructed processes, shall be verified by processes in practice. Besides, we intended to justify the (practical) need for a complex process model like the one we developed in order to approximate Pareto-optimal processes. Unfortunately, modeling of crashing and overlapping in both case studies cannot be compared with the fair assumptions in chapter 5. Thus, it did not make sense to calculate ϕ and ψ values for the Pareto-fronts of the two processes and to compare the outcomes with the insights in chapter 5. Each case study is outlined as followed. Initially, we briefly provide background information on the process itself, subsequently present the test setup including the adjustment of the process data to our model and finally discuss the optimization results.

7.1 Case Study 1: Preliminary Design Process of an Unmanned Combat Aerial Vehicle

7.1.1 Process description

The first case study comprises a process describing the preliminary design of an unmanned combat aerial vehicle (UCAV), presented in the work of Browning (1998). Generally, UCAVs are an experimental class of unmanned aerial vehicles which are designed to deliver weapons with great degree of autonomy.[24]As an example for a UCAV, Figure 7.1 depicts the Boeing X-45A UCAV, which mainly served as technology demonstrator and had its first flight on May 22, 2002.

The UCAV process, which serves as case study, is based on an actual project at The Boeing Company (Browning 1998) but data was disguised to protect company proprietary information. Motivated by the interest of the United States Air Force on some type of UCAV, Boeing initially developed some notional vehicle concepts using historical data and their perspective on Air Force needs. Subsequently, the Air Force provided more specific requirements. That prompted a kick-off of the Preliminary Design Phase, culminating in a proposal to the Air Force. For the case study, we resigned on the conceptual design phase but exclusively considered the preliminary design process. This process consisted of 14 activities (see Table 7.1) connected via 52 relationships (see Figure 7.2) and can probably not be regarded as a very complex PD process. For a more detailed description of the individual

(a) X-45A upside. (b) X-45A underside with weapons bay door open.

Figure 7.1: Picture of the Boeing X-45A UCAV, adapted from www.wikipedia.org.

[24] www.wikipedia.org

activities, e.g. input/output information, the reader may refer to Browning (1998). Despite its reduced complexity, we nevertheless feel that this design process is suited for a first case study for several reasons:

1) Preliminary design processes tend to be dynamic exhibiting much iteration which makes them interesting to investigate.
2) Instead of a long-lasting field study over several months, actual project data (in good quality) was available for the UCAV process and could be quickly adapted to our process model.
3) Given our rich process model, the scope of the process is suited as a first place to start in order to not obscure potential insights.

7.1.2 Case study setup

Process architecture

Fortunately, most of the process architecture related data necessary for our process model could be completely adopted from Browning (1998). Table 1.1 provides an overview of the 14 process activities including their cost and schedule data for the triangular distribution as well as values for the learning curve and maximal number of allowed iterations. This data is

Activities		**Time (days)**			**Costs ($k)**				
ID	**Name**	$\breve{t}_i$	$\bar{t}_i$	$\hat{t}_i$	$\breve{c}_i$	$\bar{c}_i$	$\hat{c}_i$	$\boldsymbol{L}$	$\hat{k}_i$
1	Prepare UCAV Preliminary DR&O	1.9	2	3	8.6	9	13.5	0.35	5
2	Create UCAV Preliminary Design Architecture	4.75	5	8.75	5.3	5.63	9.84	0.20	5
3	Prepare & Distribute Surfaced Models & Int. Arngmt. Drawings	2.66	2.8	4.2	3	3.15	4.73	0.60	5
4	Perform Aerodynamics Analyses & Evaluation	9	10	12.5	6.8	7.5	9.38	0.33	5
5	Create Initial Structural Geometry	14.3	15	26.3	128	135	236	0.40	5
6	Prepare Structural Geometry & Notes for FEM	9	10	11	10	11.3	12.4	1.00	5
7	Develop Structural Design Conditions	7.2	8	10	11	12	15	0.35	5
8	Perform Weights & Inertias Analyses	4.75	5	8.75	8.9	9.38	16.4	1.00	5
9	Perform S&C Analyses & Evaluation	18	20	22	20	22.5	24.8	0.25	5
10	Develop Balanced Freebody Diagrams & External Applied Loads	9.5	10	17.5	21	22.5	39.4	0.50	5
11	Establish Internal Load Distributions	14.3	15	26.3	21	22.5	39.4	0.75	5
12	Evaluate Structural Strength, Stiffness, & Life	13.5	15	18.8	41	45	56.3	0.30	5
13	Preliminary Manufacturing Planning & Analyses	30	32.5	36	214	232	257	0.28	5
14	Prepare UCAV Proposal	4.5	5	6.25	20	22.5	28.1	0.70	5

Table 7.1: Activity data for the UCAV process.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	■													
2	.4	■							.2					
3		.5	■	.4										
4	.3		.5	■										
5	.4		.5		■	.1		.1				.3	.1	
6	.1				.4	■								
7	.4					.4	■							
8						.5		■				.5		
9	.4		.5	.5				.5	■					
10				.1		.5	.2	.1		■	.4			
11						.5	.5	.5		.5	■			
12	.4					.4	.5			.5	.4	■		
13	.5				.5							.4	■	
14	.3	.4	.4	.4	.4	.4	.4	.4	.4	.4	.4	.4	.4	■

Figure 7.2: M_1 showing rework probabilities for the UCAV process**.**

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	■													
2	.5	■							.1					
3		.3	■	.5										
4	.4		.8	■										
5	.1		.1		■	.1						.3	.1	
6	.1				.3	■								
7	.5					.8	■							
8						.5		■				.5		
9	.3		.3	.3				.3	■					
10				.1		.5	.4	.3		■	.3			
11						.5	.5	.3		.3	■			
12	.5					.3	.5			.5	.5	■		
13	.9				.9							.3	■	
14	.5	.8	.8	.8	.8	.8	.8	.8	.8	.8	.8	.8	.8	■

Figure 7.3: M_2 showing rework impact values for the UCAV process.

based on interviews in the field conducted by Browning (1998) and historical data/experiences. Referring to Table 7.1, activities 5, 9 and 13 seem to be the most critical with respect to both, time and cost. Though, their learning curve value is rather low which means that rework of those activities is expected to be worked off quickly and cheap. Notably, learning does not occur at all for activities 6 and 8 hence constituting potential bottleneck activities. In addition to the activity data, information flows between the activities (Figure 7.2), i.e. M_1 in our model, as well as rework impact information (Figure 7.3), i.e. M_2, were available in the study of Browning (1998) and completely adopted.

Crashing and overlapping

Since the process model in Browning (1998) did not account for overlapping and crashing, we had to collect the respective data interviewing an aerospace & aeronautics expert who was also familiar with the UCAV process. Therefore, we like to point out that the proposed crashing data (Table 7.2) describes no actual data but merely proxies based on (subjective) knowledge/experience in the field. According to this expert, each activity could be theoretically crashed (Table 7.2) through the assignment of additional staff hence provoking multiple payment cost for crashing (see section 4.4). Extra costs for crashing were calculated

ID	Activity Name	Continuous or Discrete?	Intensity (α)	Maximum Crashing	Resource
1	Prepare UCAV Preliminary DR&O	C	Medium = 6	25%	Staff
2	Create UCAV Preliminary Design Architecture	C	High = 12	25%	Staff
3	Prepare & Distribute Surfaced Models & Int. Arngmt. Drawings	C	Medium = 6	10%	Staff
4	Perform Aerodynamics Analyses & Evaluation	C	High = 12	25%	Staff
5	Create Initial Structural Geometry	C	High = 12	25%	Staff
6	Prepare Structural Geometry & Notes for FEM	C	High = 12	50%	Staff
7	Develop Structural Design Conditions	C	Medium = 6	25%	Staff
8	Perform Weights & Inertias Analyses	C	Medium = 6	25%	Staff
9	Perform S&C Analyses & Evaluation	C	High = 12	25%	Staff
10	Develop Balanced Freebody Diagrams & Ext. Applied Loads	C	High = 12	25%	Staff
11	Establish Internal Load Distributions	C	High = 12	10%	Staff
12	Evaluate Structural Strength, Stiffness, & Life	C	High = 12	10%	Staff
13	Preliminary Manufacturing Planning & Analyses	C	Medium = 6	50%	Staff
14	Prepare UCAV Proposal	C	Medium = 6	10%	Staff

Table 7.2: Crashing related data for the UCAV process.

for all activities using the continuous crashing function as defined by equation 4.19. Thereby, no activity is supposed to be elongated through the crashing resources and hence κ_i=1 holds. Importantly, the most cost- and time intensive activity 13 may be crashed by 50%. Considering the fact that extra cost for crashing this activity are expected to be moderate due to a medium high α value, we suggest a high positive impact of crashing activity 13 on entire process duration at a relatively low price.

Analogical to crashing, we developed the overlapping information necessary on the basis of interviews with the aforementioned expert. The result, primarily matrices M_3-M_6, is depicted at Figure 7.4 through Figure 7.7. Clearly, overlapping intensities are not very high

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	■													
2	1	■							1					
3		1	■	1										
4	1		1	■										
5	1		1		■	1		1				1	1	
6	1				1	■								
7	1					1	■							
8						1		■				1		
9	1		1	1				1	■					
10				1		1	1	1		■	1			
11						1	1	1		1	■			
12	1					1	1			1	1	■		
13	1				1							1	■	
14	1	1	1	1	1	1	1	1	1	1	1	1	1	■

Figure 7.4: M_3 showing maximal overlapping available values for UCAV process.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	■													
2	0	■							0					
3		0	■	0										
4	0		0	■										
5	0		0		■	0		0				0	0	
6	0				0	■								
7	0					0	■							
8						0		■				0		
9	0		0	0				0	■					
10				0		0	0	0		■	0			
11						0	0	0		0	■			
12	0					0	0			0	0	■		
13	0				0							0	■	
14	0	0	0	0	0	0	0	0	0	0	0	0	0	■

Figure 7.5: M_4 showing minimal overlapping needed values for UCAV process.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	■													
2	1	■							1					
3		1	■	1										
4	1		1	■										
5	1		1		■	1		1				1	1	
6	1				1	■								
7	1					1	■							
8						1		■				1		
9	1		1	1				1	■					
10				1		.8	1	1		■	1			
11						.8	1	1		1	■			
12	1					1	1			1	1	■		
13	1				1							1	■	
14	1	1	1	1	1	1	1	1	1	1	1	1	1	■

Figure 7.6: M_5 showing minimal overlapping available values for UCAV process.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	■													
2	**0**	■							.8					
3		**0**	■	.5										
4	.2		**0**	■										
5	**0**		.2		■	.9		.3				.5	.8	
6	.2				.1	■								
7	**0**					**0**	■							
8						**0**		■				.5		
9	**0**		**0**	.2				.2	■					
10				.5		**0**	.5	.1		■	.5			
11						**0**	.1	**0**		.5	■			
12	**0**					**0**	**0**			.2	.2	■		
13	**0**				**0**							.2	■	
14	**0**	**0**	.3	.3	.3	.3	.3	.3	.3	.3	.3	.3	.3	■

Figure 7.7: M_6 showing maximal overlapping needed values for UCAV process.

while several firm predecessor constraints (see M_6) exist. Hence, we assume, contrary to crashing, only a minor impact of overlapping on overall process cost and time. In the event of overlapping between sequentially linked/coupled activities, we selected the linear overlapping function (equation 4.12) to model the rework penalties. As rework impact and rework probabilities can be regarded as some kind of indicator for the amount of rework due to (faulty) preliminary information input, we set $\alpha_{ij} = m_{1,ij} \cdot m_{2,ij}$ for the overlapping function.

GA and simulation settings

Finally, we propose the parameters for the optimization and process simulation. Generally, the optimal determination of all ε-MOEA parameters constitutes an optimization problem by itself. To obtain a good and wide-spread Pareto-front we thus determined most of the parameters by empirical tests and prior experiences with GA/MOEA related problems rather than using mathematical equations.

We used a relatively high population of 10.000 chromosomes, encoded with n_B=8 bits, as computational time was no concern and let the ε-MOEA run maximal 8.000 generations. Though, the ε-MOEA could be finished earlier if the number of solutions in the archive population did not change for 1.500 generations indicating a stall very close to the true

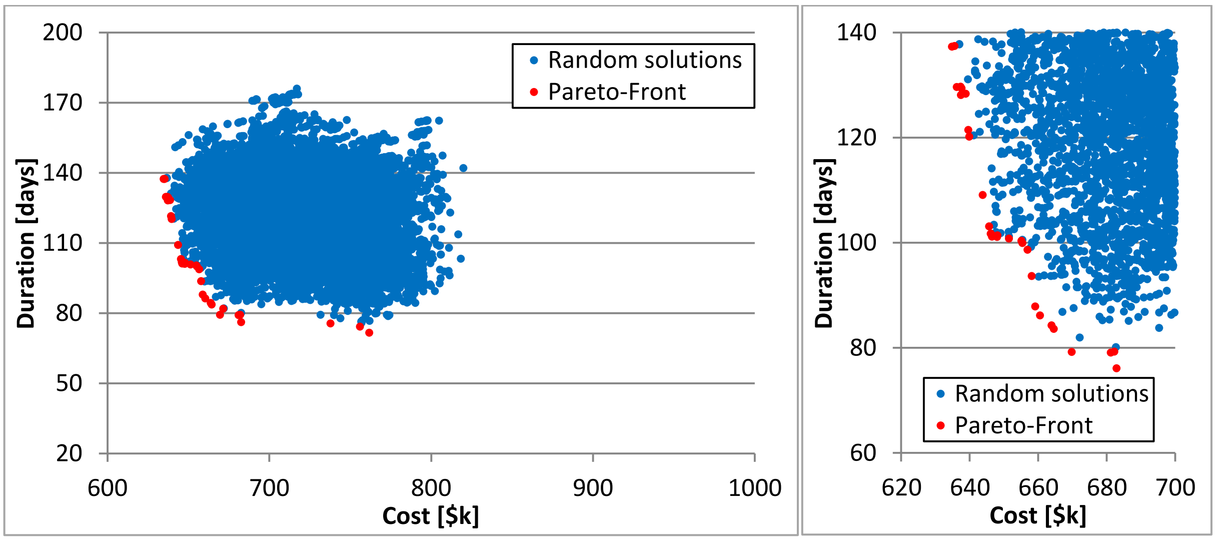
Pareto-front. Due to Meier et al. (2007) we pushed a global search and set the crossover probability p_c to 100% while mutation probability p_m was set to a low value of 0.25 for each of the four chromosome segments (i.e. mutation is expected to occur for each segment with probability 25%).

The discrete event process simulation, discussed in chapter 4, served as objective/fitness function for the ε-MOEA. Thereby, each process was simulated 1.500 runs and the average values for cost and duration were assigned to the corresponding chromosome as fitness values. Since the high number of simulation runs yielded relatively stable outputs with only minor deviations for identical processes, we set the size of the two-dimensional hyper-boxes to 0.2, i.e. ε=0.2 holds for the ε-MOEA. In this way, we expect ε to be small enough to generate a sufficiently high number of Pareto-solutions.

7.1.3 Optimization results

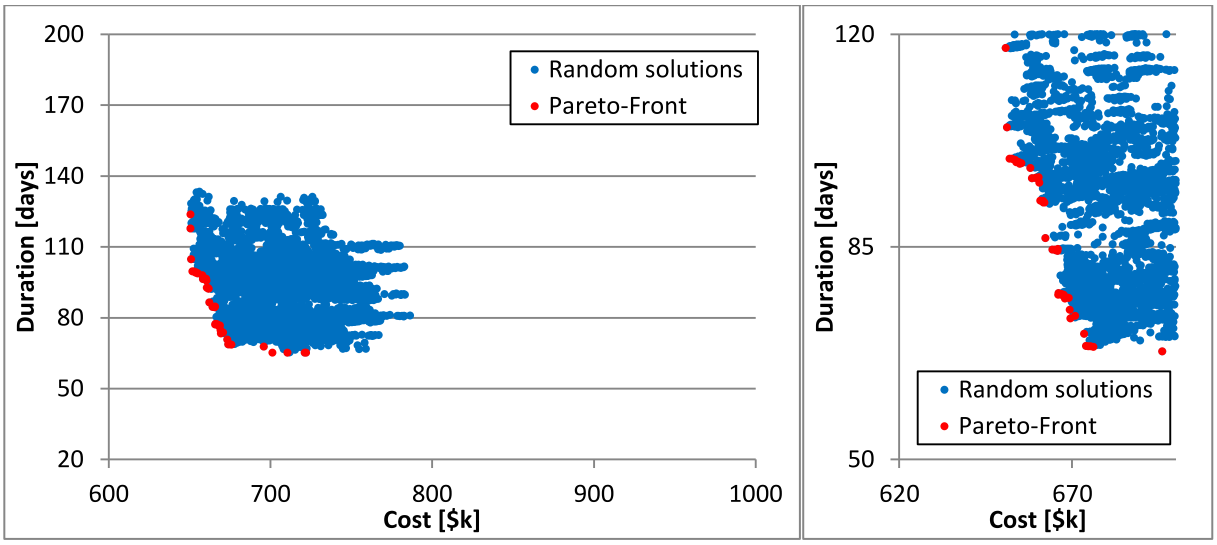
Comparison of Pareto-fronts for work policies P_1-P_5

Assuming the case study setup, one complete run of the ε-MOEA for the UCAV process last approximately 45 minutes on a workstation featuring 6GB RAM, 12 virtual cores at 3.33 GHz clock rate and Windows 7 as operating system. First of all, we were interested in the outcomes of the Pareto-front for all five work policies which are separately displayed at Figure 7.8-Figure 7.12. In addition to the Pareto-fronts, all figures show the simulated time-cost results for 10.000 randomly generated chromosomes/processes (a comparison of all work policies is shown at Figure 7.13) in order to demonstrate the difference between “guessing” and sophisticated optimization strategy. Interestingly, this discrepancy is not very salient for all work policies because of the high number of random solutions and low complexity of the process: 1) with respect to process architecture, merely 14 activities exist and cannot be sequenced in any arbitrary way because of predecessor constraints 2) the number of potential overlapping scenarios is limited because of low overlapping intensities (see matrices M_3-M_6).



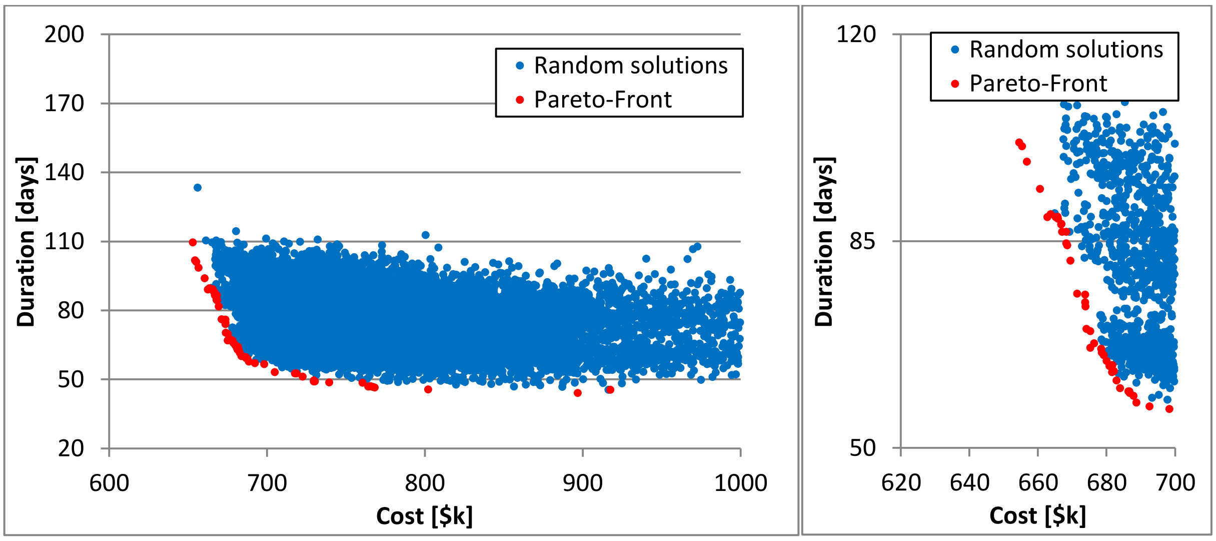
(a) Overall objective space (b) Pareto region

Figure 7.8: Pareto-front and random solutions for the UCAV process assuming work policy P_1.



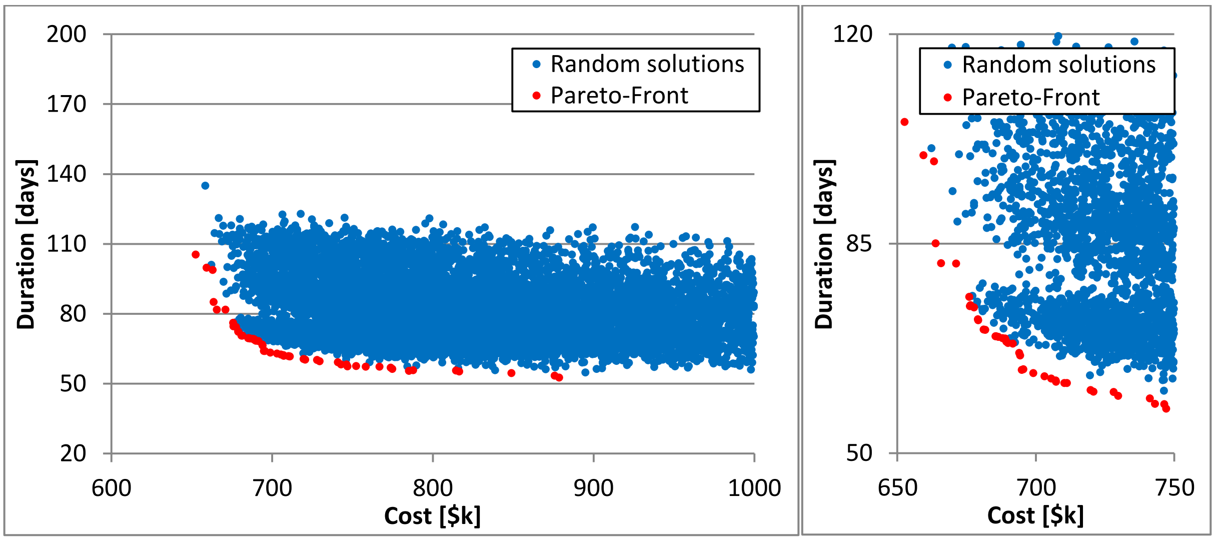
(a) Overall objective space (b) Pareto region

Figure 7.9: Pareto-front and random solutions for the UCAV process assuming work policy P_2.



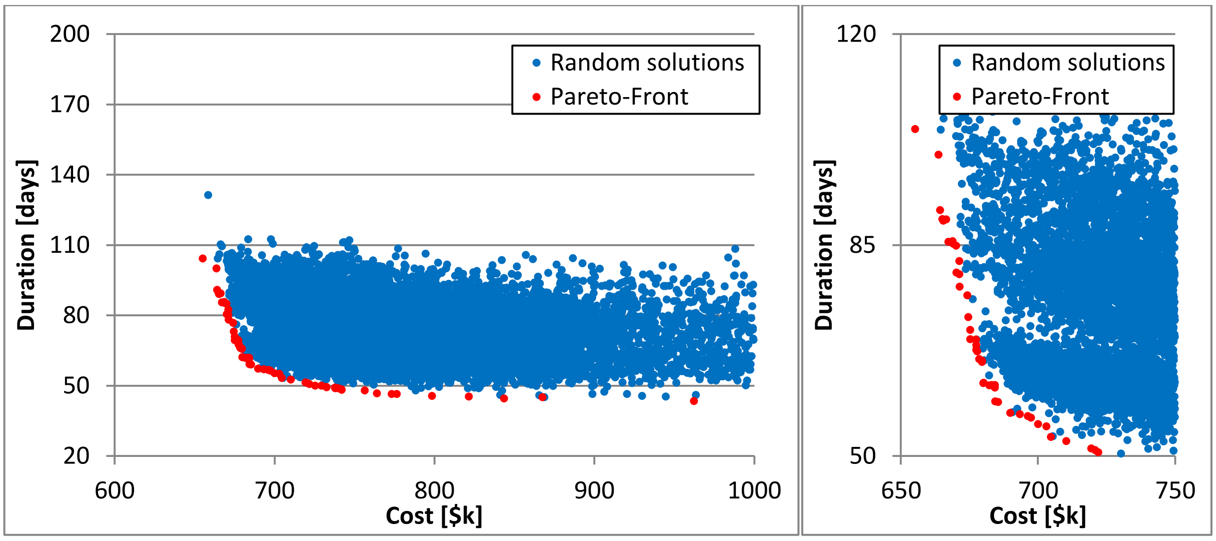
(a) Overall objective space (b) Pareto region

Figure 7.10: Pareto-front and random solutions for the UCAV process assuming work policy P_3



(a) Overall objective space (b) Pareto region

Figure 7.11: Pareto-front and random solutions for the UCAV process assuming work policy P_4.



(a) Overall objective space (b) Pareto region

Figure 7.12: Pareto-front and random solutions for the UCAV process assuming work policy P_5.

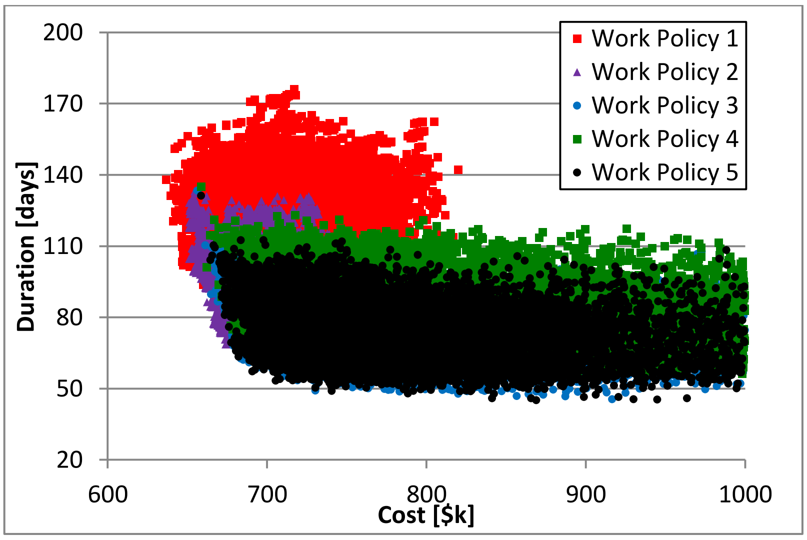


Figure 7.13: Comparison of random solutions for the UCAV process assuming work policies P_1-P_5.

Nevertheless, the five diagrams let us already suspect that the number of feasible processes is much higher than the number of actual Pareto-optimal processes. Consequently, the chance for managers to select a suboptimal process is very high. For instance, even the min-max values of the random solutions for cost and duration assuming the less complex work policies P_1 and P_2 range from \$636k-\$820k and from 75days to 175days, respectively. With more complexity, i.e. crashing and overlapping, we note an actually greater min-max interval for work policies P_3-P_5, namely \$653k-\$1.450k and 43days-133days. Consequently, the effort for a thorough process planning, including modeling, simulation and optimization, may significantly pay off compared to a pure "guess".

Analysis of the overall Pareto-front

Plotting the Pareto-fronts of all five work policies into one diagram yields Figure 7.14 and ranges from \$635k to \$962k with respect to cost and from 137 days to 43 days, respectively. Thus, the percental volatility in terms of duration is higher than in case of cost. Besides, the Pareto-fronts confirm some of our previously derived insights in chapter 5 regarding the time-cost scale-up for P_1-P_5. Although the absolute difference between the five Pareto-fronts are not rather salient (because of the aforementioned low process complexity), we nevertheless

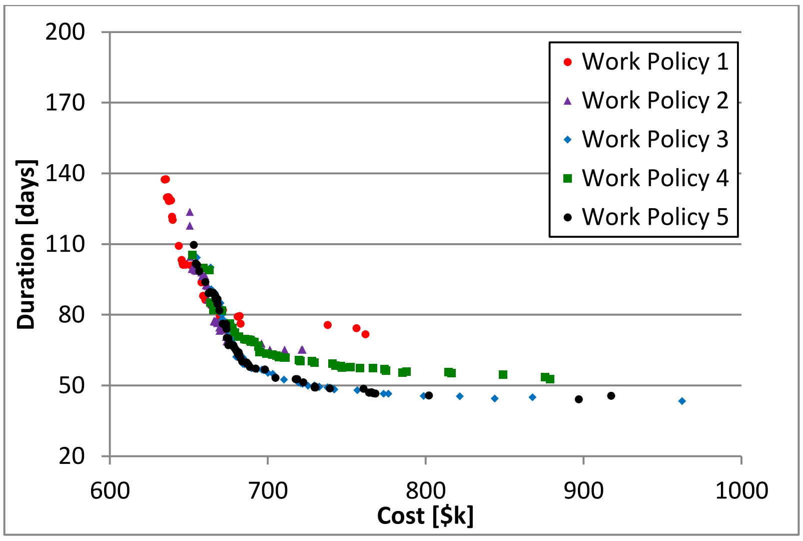


Figure 7.14: Comparison of Pareto-fronts for the UCAV process assuming work policies P_1-P_5.

note outcomes which are consistent with Figure 5.4 and Figure 5.6 describing the scale-up for time-and cost assuming different n_{fb}/n_{ff} combinations: P_1 generated the cheapest but also slowest processes, applying P_2 and P_4 resulted in medium costly and slow processes while P_3 and P_5 provoked the quickest and most expensive processes. Neglecting crashing and overlapping for P_3-P_5 indeed changes absolute values of the corresponding Pareto-fronts but not this basic statement as illustrated by Figure E.19 in the appendix. The second case study will highlight the different scale-up behaviors for P_1-P_5 much more as it involves more activities and (feedback) relationships.

Composing the overall Pareto-front for the UCAV process out of the five individual Pareto-fronts for every work policy allows us to analyze the “global” Pareto-optimal processes more in detail. Figure 7.15 depicts this overall efficient frontier, decomposed into four different sectors containing processes with diverse time-cost characteristics: the most cost and resource conserving processes can be found in sector 1, sector 2 includes speedier processes without the application of crashing & overlapping whereas sector 3 represents the probably most balanced (in terms of time and cost) Pareto-processes with the use of crashing and overlapping, and finally sector 4 comprises the fastest but most cost-intensive processes. Evidently, the sectors containing cheap processes feature an almost vertical spread/volatility

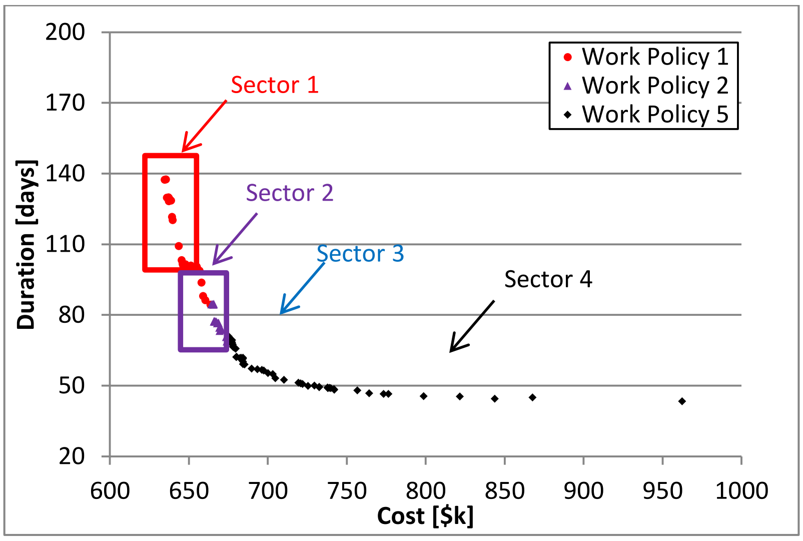


Figure 7.15: Overall Pareto-front for the UCAV process assuming work policies P_1-P_5 with four disjoint sectors which are analyzed more in detail.

Parameter	**Sector 1**	**Sector 2**	**Sector 3**	**Sector 4**
Dominating work policy	P_1	P_2	P_5	P_5
Average number of feedback relationships	10.2	16.3	16.9	22.4
Average distance of a feedback mark to diagonal	4.43	4.06	4.07	4.47
Weighted average distance of a feedback mark to diagonal	1.07	1.23	1.25	1.33
Average number reworks	2.77	7.10	7.82	10.2
Average number partial reworks	2.61	4.07	5.43	8.12
Average crashing intensity (1st iteration) for any activity (min: 0, max: 0.24)	0	0	0.09	0.14
Average crashing intensity (1st iteration) for critical activities 5, 9 and 13 (min: 0, max: 0.33)	0	0	0.10	0.19
Average crashing intensity (1st iteration) for most critical activity 13 (min: 0, max: 0.50)	0	0	0.15	0.40
Average intensity for overlapping available (1st iteration) between two activities (min: 0.8, max: 1.0)	0	0	0.91	0.89
Average overlapping received intensity (1st iteration) between two activities (min: 0, max: 0.38)	0	0	0.21	0.18

Table 7.3: Selected process parameters for four disjoint sectors of the overall Pareto-front.

in terms of duration while the sector 4 with the fastest processes is spread-out horizontally and shows only little improvement of duration at a very high prize. Importantly, the shape of the combined Pareto-front is strongly continuous thus confirming our previous simulation results in chapter 5 for artificial processes with low n_{fb} (see upper right corner of Figure 5.14). An evaluation of each sector, presented in Table 7.3, revealed substantial differences in process characteristics:

1. Although exceptions may exist, we note a certain correlation between the number of feedbacks and process cost/schedule: the more feedbacks the higher cost and the more rapid the process. Vice versa, cost dropped and processes became slower with less feedback marks. Obviously, in average, processes were accelerated at the expense of more parallel activities (iterative overlapping) which provoke a higher amount of costly rework (and partial rework). Contrary, we could not find a correlation between the placement of feedback relationships, in particular their distance to the diagonal (indicating the "length" of the feedback loop) with or without feedback probability as weight, and process/duration.

	1	2	3	5	4	6	10	7	8	11	9	12	13	14
1	■													
2	.4	■							.2					
3		.5	■	.4										
5	.4		.5	■		.1			.1			.3	.1	
4	.3		.5		■									
6	.1			.4		■								
10					.1	.5	■	.2		.4				
7	.4					.4		■						
8						.5			■			.5		
11						.5	.5	.5	.5	■				
9	.4		.5		.5				.5		■			
12	.4					.4	.5	.5		.4		■		
13	.5			.5							.4		■	
14	.3	.4	.4	.4	.4	.4	.4	.4	.4	.4	.4	.4	.4	■

Figure 7.16: M_1 for a cheap but slow process.

	6	10	1	2	14	5	3	4	9	13	8	7	12	11
6	■ .31		🟩 .1			🟩 .4								
10	🟩 .5	■ .16						🟩 .1			🟩 .1	🟩 .2		🟩 .4
1			■ .04											
2			🟩 .4	■ .23					🟨 .2					
14	🟩 .4	🟩 .4	🟩 .3	🟩 .4	■ .01	🟩 .4	🟨 .4	🟩 .4	🟨 .4	🟨 .4	🟩 .4	🟩 .4	🟨 .4	🟩 .4
5	🟨 .1		🟩 .4			■ .10	🟩 .5			🟩 .1	🟨 .1		🟩 .3	
3				🟩 .5			■ .06	🟥 .4						
4			🟨 .3				🟩 .5	■ .18						
9			🟩 .4				🟩 .5	🟩 .5	■ .18		🟩 .5			
13			🟩 .5			🟩 .5				■ .47			🟩 .4	
8	🟩 .5										■ .18		🟨 .5	
7	🟩 .4		🟩 .4									■ .10		
12	🟩 .4	🟩 .5	🟩 .4									🟩 .5	■ .07	🟩 .4
11	🟩 .5	🟩 .5									🟩 .5	🟩 .5		■ .05

Figure 7.17: M_1 for an expensive but fast process.

2. While overlapping intensities do not greatly differ for sectors 3 and 4, quick processes are crashed at a higher rate, especially the aforementioned critical activities 5, 9 and 13.

Both of the above observations are visualized by Figure 7.16 and Figure 7.17 which represent the DSMs M_1 for both, a cheap but slow process (Figure 7.18) and an expensive but fast process (Figure 7.17). In addition to the process architecture, the DSM in Figure 7.17 furthermore involves data regarding crashing intensities in the 1st iteration for any activity (white numbers on the diagonal). Besides, background colors for the DSM cells indicate the overlapping intensities in the 1st iteration (M_6) for all relationships: green colored cells denote an intensity $m_{6,ij}<0.2$, yellow colored cells intensities $0.2<m_{6,ij}<0.4$ and finally red colored cells represent high overlapping intensities $m_{6,ij}>0.4$.

Referring to the activity sequence of a fast process (Figure 7.17), it is interesting to note that activities 6 and 10 precede the requirements and objectives and that activity 14 is executed before all activities have been finished. Still, it is a feasible sequence as all firm predecessor constraints are satisfied. The position of these three activities merely indicates that, in order to expedite the process, their initial execution should begin earlier while their rework may occur later.

Justification of model complexity

Another contribution of the optimization results is the justification to build, and use, a complex PD process model like the one we introduced in chapter 4. Collectively allowing cyclic process architectures, crashing, overlapping as well as different work policies explicitly affects the objective space for potential processes as demonstrated by Figure 7.18. Assuming work policy P_3, this chart more or less displays the separate Pareto-fronts for different levels of model complexity:

- Only changes in the process architecture are allowed but not application of overlapping or crashing (red dots in Figure 7.18).
- Assuming constant process architecture (the original one) and no crashing, only changes of the overlapping intensities are permitted (blue dots in Figure 7.18).
- Assuming constant process architecture (the original one) and no overlapping between sequentially linked/coupled activities, only changes of the overlapping intensities are permitted (green dots in Figure 7.18).

Analogical diagrams for work policies P_4 and P_5 can be found in the appendix, Figure E.13-Figure E.18. The differences between all four Pareto-fronts are enormous while changing the process architecture seems to be the greatest lever for the UCAV process. However, as

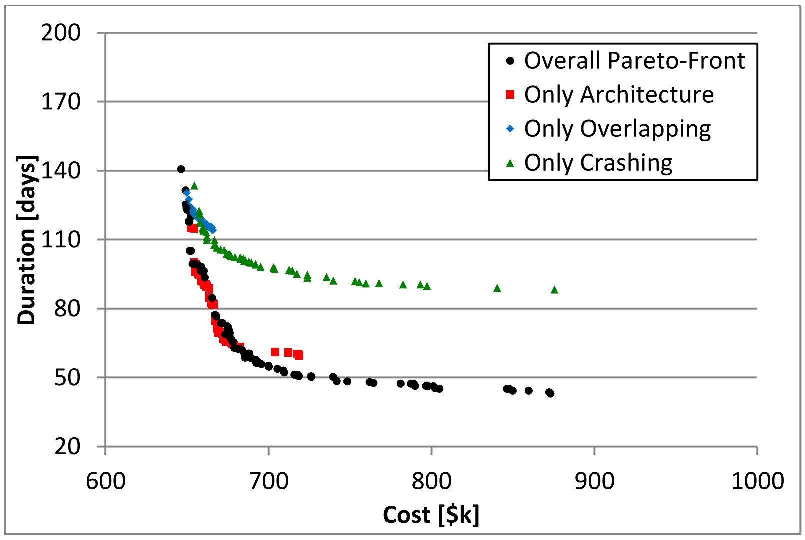


Figure 7.18: Comparison of the overall Pareto-front (all work policies) for the UCAV process with the Pareto-fronts assuming work policy P_3 for different levels of complexity.

conclusion, only the combined consideration of all relevant process parameters reveals the best (Pareto-optimal) solutions.

7.2 Case study 2: Development process of an automotive hood

7.2.1 Process description

Indeed, the UCAV case study produced some important contributions: 1) to some extent it already verified our assumptions on time-cost tradeoffs due to work policy rules which were initially based on artificial processes and 2) it demonstrated that a certain model complexity is necessary to identify global Pareto-solutions. Though, the UCAV process itself consisted only of a few activities barely connected with each other. Therefore, we feel that it is not perfectly suited to fully demonstrate the impact of process parameters, particularly the effects of iteration, on time-cost tradeoffs. Hence, we analyzed a second, more complex, process which is based on an actual project in practice, too. A crucial factor for the selection of this process was once more the large set of available process data.

This time, however, we examine the PD process of an automotive hood at The Ford Company. Consistent with the UCAV case study, data is realistic but disguised to protect company proprietary information. Figure 7.19 shows the components of a generic hood-system thereby highlighting that the entire system is composed of several individual

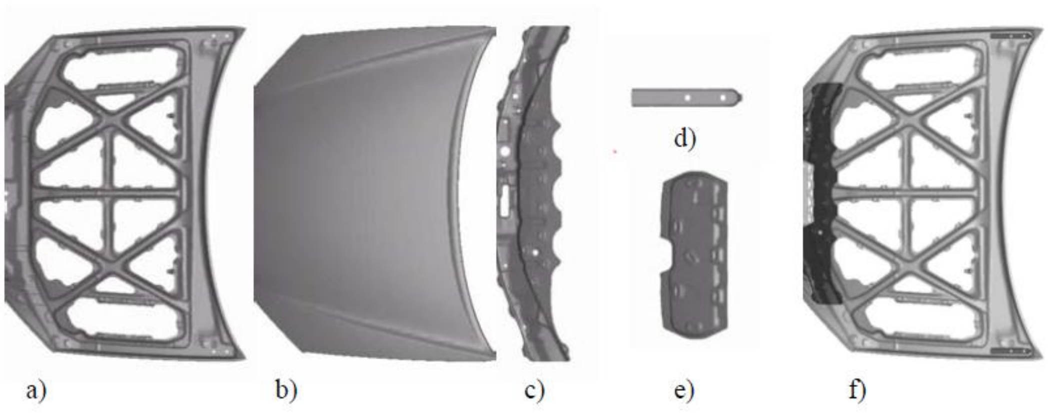


Figure 7.19: Components of a generic hood subsystem, adopted from Galbraight et. al (2003); a) hood inner b) hood outer c) main reinforcement d) hood ringe reinforcement e) latch reinforcement f) assembly without the outer panel.

components. Basically, the process comprises 43 activities connected via 232 relationships and considers not only the development of hood components but also the development of tooling for assembly and stamping (see Table 7.4). Also worth mentioning is the occurrence of several verification & validation activities potentially provoking feedback loops.

7.2.2 Case study setup

In the following, we present only partial material of the entire case study setup due to the size of the DSMs prescribing process architecture as well as crashing- and overlapping data. Though, the entire set of data, consistent to the UCAV case study, can be found in section F of the appendix.

Process architecture

The process architecture related data for the hood development is based on a study of Zambito (2000) at The Ford Company and partially presented in Yassine et. al (2000). Again, data gathering in the field was mostly conducted through interviews with engineers involved in the project. A brief description of each activity as well as its respective time, cost and learning values can be found in tabular form at Table 7.4. The corresponding pattern of interaction between the individual activities, i.e. M_1, is depicted at Figure 7.20 while M_2 is placed in the appendix (Figure F.21). Evidently, the activities referring to the development of tooling (i.e. activities 26, 27 and 28) as well as activity 32 can be regarded as the most critical activities. Beside their above average cost and duration values, they feature only minor improvement effects through learning (=high values in vector $\boldsymbol{L}$) hence making their rework expensive and long-lasting. Notably is also the high amount of firm predecessor constraints (see Figure F.25). Beginning with activity 16, almost all succeeding activities are subject of predecessor constraints. Thus, we expect only marginal differences in process time and cost for varying process architectures.

Activities		Time (days)			Costs ($k)				
ID	**Name**	$\breve{t}_i$	$\bar{t}_i$	$\hat{t}_i$	$\breve{c}_i$	$\bar{c}_i$	$\hat{c}_i$	L	$\hat{k}_i$
1	Strategies for product, mkt, mfg, supply, design, and reusability confirmed (Est. PDL)	0	0	0	0	0	0	-	-
2	Select powertrain lineup	0	0	0	0	0	0	-	-
3	Select materials for all system components	13.5	15	16.5	10	11.3	12.4	0.75	5
4	Freeze proportions and selected hard-points	54	60	66	41	45	49.5	0.75	5
5	Verify that hard-points and structural joint designs are compatible w/ program targets	36	40	44	27	30	33	0.75	5
6	Approve master sections	36	40	44	41	45	49.5	0.85	5
7	Develop initial design concept (preliminary CAD model)	36	40	44	68	75	82.5	0.10	5
8	Estimate blank size	0.9	1	1.1	0.7	0.75	0.8	0.10	5
9	Estimate efforts	0.9	1	1.1	4.1	4.5	4.9	0.10	5
10	Develop initial attachment scheme	4.5	5	5.5	3.4	3.8	4.13	0.50	5
11	Estimate latch loads	4.5	5	5.5	20	22.5	24.8	0.10	5
12	Cheat outer panel surface	9	10	11	10	11.3	12.4	0.50	5
13	Define hinge concept	18	20	22	20	22.5	24.8	0.50	5
14	Get prelim. mfg and asy feas. (form, holes, hem, weld patterns, mastic locations, adhesive)	4.5	5	5.5	3.4	3.75	4.13	0.50	5
15	Perform cost analysis (variable and investment)	1.8	2	2.2	16	18	19.8	0.50	5
16	Perform swing study	1.8	2	2.2	2	2.3	2.5	0.75	5
17	Theme approval for interior and exterior appearance (prelim surf available)	13.5	15	16.5	20	22.5	24.8	0.10	5
18	Marketing commits to net revenue; initial ordering guide	4.5	5	5.5	8.4	9.4	10.3	0.10	5
19	Program DVPs and FMEAs complete	9	10	11	10	11.3	12.4	0.75	5
20	Approved theme refined for craftsmanship execution (consistent w/ PA objectives)	13.5	15	16.5	30	33.8	37.1	0.10	5
21	PDN0 - Interior and exterior Class 1A surfaces transferred to engineering (+/- 0.5mm)	2.7	3	3.3	4.1	4.5	5	0.10	5
22	Conduct cube review and get surface buyoff	18	20	22	54	60	66	0.25	5
23	Verify mfg and asy feas. (form, holes, hem, weld patterns, mastic locations, adhesive)	9	10	11	64	71	78	0.75	5
24	Evaluate functional performance (analytically)	13.5	15	16.5	81	90	99	0.50	5
25	PDN 1 - Release system design intent level concept to manufacturing	18	20	22	122	135	149	0.50	5
26	Develop stamping tooling	378	420	462	2835	3150	3465	0.90	5
27	Develop hemming tooling (if applicable)	57.6	64	70.4	475	528	581	0.75	5
28	Develop assembly tooling	90	100	110	810	900	990	0.75	5
29	PDN2 - Last Class 1surface verified and released for major formed parts	18	20	22	176	195	215	0.50	5
30	PDN3 - Final math 1, 2, & 3 data released	18	20	22	189	210	231	0.50	5
31	CAD files reflect pre-CP verification changes	18	20	22	203	225	248	0.75	5
32	Make "like production" part and asy tools / ergonomics / process sheets (to extent feasible)	72	80	88	864	960	1056	0.75	5
33	First CPs available for tuning and durability testing	4.5	5	5.5	57.3	63.8	70.1	1.00	5
34	Complete CMM analysis of all end items & subassemblies	9	10	11	122	135	149	0.10	5
35	Perform DV tests (physical)	18	20	22	257	285	314	0.10	5
36	Verify manufacturing and assembly process capability	4.5	5	5.5	67.5	75	82.5	0.10	5
37	Complete prelim. ESO for: CP durability testing	13.5	15	16.5	213	236	260	0.10	5
38	Complete prelim. ESO for: Initial set of road tests completed	27	30	33	446	495	545	0.10	5
39	Complete prelim. ESO for: Known changes from CP containable for 1PP	4.5	5	5.5	77.6	86.3	94.9	0.10	5
40	Complete prelim. ESO for: Design is J1 level - no further changes except No-Blds	4.5	5	5.5	81	90	99	0.10	5
41	Supplier commitment to support 1PP w/ PSW parts	2.7	3	3.3	50.6	56.3	61.9	0.10	5
42	Complete prelim. ESO for: Eng. confidence that objectives will be met declared	2.7	3	3.3	52.7	58.5	64.4	0.10	5
43	Readiness to proceed to tool tryout (TTO), 1PP and Job #1	9	10	11	182	203	223	0.10	5

Table 7.4: Activity data for the hood development process.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43
1	■																																									
2		■																																								
3			■					.4	.4						.4		.1	.3						.1																		
4		.1		■	.4	.3				.4																																
5				.3	■	.3	.6																																			
6	.4			.4	.4	■	.3																																			
7	.4				.4	.4	■			.4		.2	.3			.4	.2			.4	.4			.4																		
8						.3	.4	■					.3				.3			.2	.1																					
9	.4		.4						■				.1				.1																									
10	.3		.1			.4	.4			■			.4				.3			.2	.1			.1																		
11						.4	.6				■						.1				.1			.1																		
12			.4				.4				.2	■									.3																					
13	.3					.3							■			.1				.2																						
14						.3	.4	.4		.4			.3	■			.1			.2																						
15	.4		.4			.4	.4	.3		.2			.3	.3	■		.3	.3		.2																						
16						.3							.3			■				.4																						
17	.3					.3								.3		.1	■			.2																						
18																	.3	■																								
19	.3		.3				.4			.2			.3						■																							
20			.3														.3			■																						
21			.3				.2									.2	.4			.4	■	.4	.3																			
22			.3			.3											.1			.2	.4	■	.3																			
23	.3					.4	.6	.3		.4			.3	.4			.4			.2	.3	.2	■			.3	.3	.3														
24			.4				.3			.2			.3	.3		.1	.3		.1	.2	.3	.2	.3	■																		
25										.6			.4	.4			.4			.4	.4	.6	.4	.2	■				.2	.4	.4											
26						.4															.4		.4	.2	.4	■			.1	.1	.4											
27						.4															.4	.2	.4		.3		■	.1		.1	.4											
28						.4															.4		.4		.3		.1	■		.1	.4											
29																				.6		.6	.3		.4				■													
30																							.4						.2	■	.4				.6							.1	
31	.4																						.4		.4				.2		■											.1	
32																						.2	.4		.4				.2	.4	.3	■											
33																																.6	■	.1									
34																																	.4	■									
35																			.2														.4	.2	■								
36																																.6	.4	.2		■							
37																																		.1	.4	.1	■						
38																																	.4					■					
39																																			.4	.1	.3	.3	■	.1	.2	.1	
40																													.2	.4	.3				.6	.3	.3	.4	.4	■			
41																																							.4	.4	■		
42																																			.6	.3	.3	.4	.3			■	
43																																		.1	.6	.4	.3	.3	.4	.4	.1	.1	■

Figure 7.20: DSM M_1 for the hood development process.

Crashing and overlapping

Since the process models in the studies of Zambito (2000) and Yassine et. al (2000) did not account for crashing or overlapping, we had to ask an expert familiar with the hood development process for the missing information. According to the input of the expert, we generated Table F.112 focusing on crashing as well as the overlapping related matrices M_3-M_6 (Figure F.22 through Figure F.25 in the appendix). Notably, crashing for the hood process is, opposed to the UCAV process, not completely staff driven. In lieu, some of the aforementioned critical activities may be crashed through the purchase of additional machinery – though, at a very high prize. Finally, we assume κ_i=1 holds for all activities.

In terms of overlapping two sequentially connected activities, matrices M_5 and M_6 of the hood development process highlight that the possibilities to actually overlap are rather limited. Activities 16-43 require almost perfect input information from their predecessors in order to begin execution. Therefore, the corresponding cells in M_6 feature low values while M_5 values are high suggesting that overlapping durations will be low. As an exception, the early activities 1-15 are less constrained thus permitting a considerable amount of overlapping between a few activity pairs. All in all, however, we do not expect a great impact of overlapping on the Pareto-front for this case study. Identical to the first case study, rework penalty in case of overlapping was calculated according to the linear overlapping function (equation 4.12) and set $\alpha_{ij} = m_{1,ij} \cdot m_{2,ij}$.

GA and simulation

Despite a three times greater number of activities and almost five times more relationships, we did not have to dramatically change the settings for the ε–MOEA and simulation because of the outlined predecessor constraints. Stable simulation outputs for the average cost and duration values of a chromosome could be obtained after 1.800 simulation runs (empirically determined). Regarding the optimization parameters, additional constraints may cut down the size of the feasible search space but increase the probability of generating redundant solutions (due to our repair mechanism) on the other hand. Thus, we increased the number of generations while population size remained identical to the UCAV process. Explicitly, the ε–MOEA was run with 10.000 8-bit encoded chromosomes for maximal 15.000 generations.

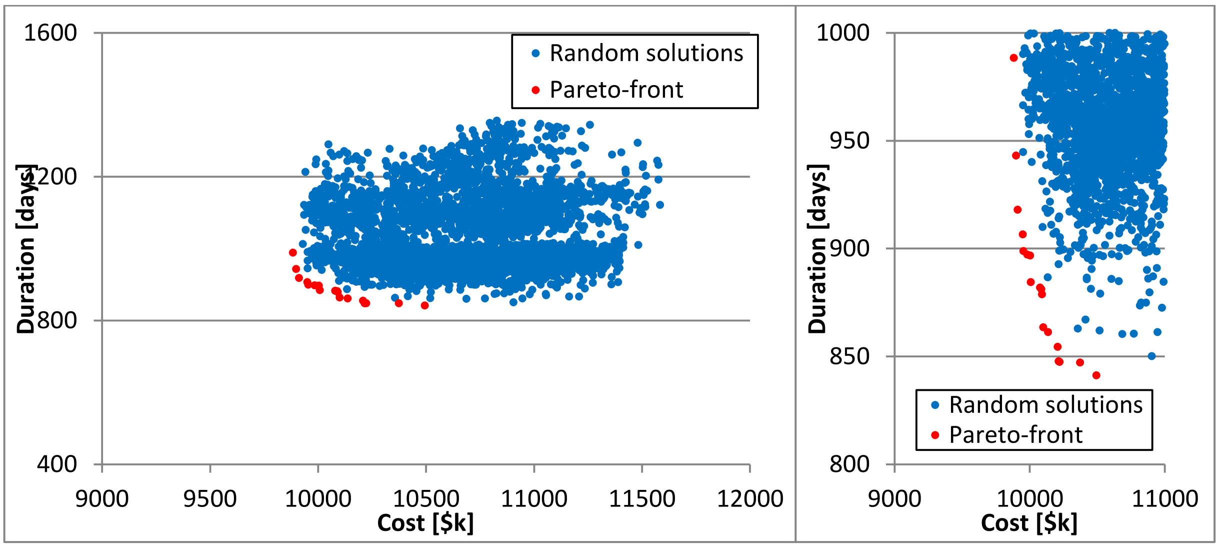
7.2.3 Optimization results

Comparison of Pareto-fronts for work policies P_1-P_5

To remain consistent with the first case study, we conducted equal tests for the hood development process proposing accordant charts in the following. First of all, Figure 7.21-Figure 7.25 visualize the Pareto-fronts and 10.000 randomly generated process solutions for each of the five work policies. Similar to the UCAV process, the time-cost differences between Pareto- optimal solutions and random solutions are not very salient. Yet, this outcome is not very surprising as the high number of predecessor constraints was expected to limit the impact of modifications in the process architecture, which might be the greatest lever of all process parameters, for each policy. Nevertheless, a certain gap exists and volatility of min-max cost & duration values is clearly higher than in case of the UCAV process. Therefore, the optimization results insistently recommend the use of a sophisticated optimization strategy instead of a random walk.

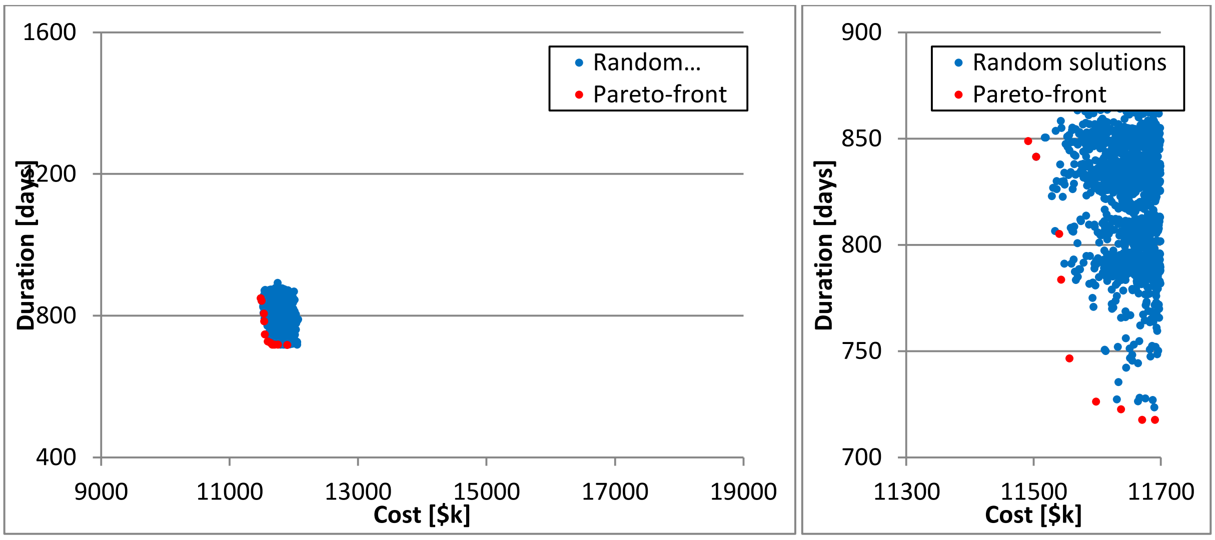
Analysis of the overall Pareto-front

A comparison of random solutions, which can be regarded as an indicator of the entire objective spaces, between all work policies is shown at Figure 7.26. In contrast, Figure 7.27 displays the outcome of randomly generated processes for all work policies assuming that overlapping and crashing are prohibited. In this way, we can compare the results with the theoretically approximated objective spaces in section 5.1.4 (see Figure 5.14). Definitely, we note significant differences between the data plotted at Figure 7.26/Figure 7.27 and the respective charts for the UCAV process (Figure 7.13 and Figure E.20). Apparently, the higher process dynamics of the hood development process, compared to the UCAV process, emphasizes much more the impact of work policy rules and crashing/overlapping on process time & cost. In fact, the outputs look similar to the analogical test results with artificial processes in section 5.1.3. Also, Figure 7.27 verifies our proposition in section 4.6 regarding



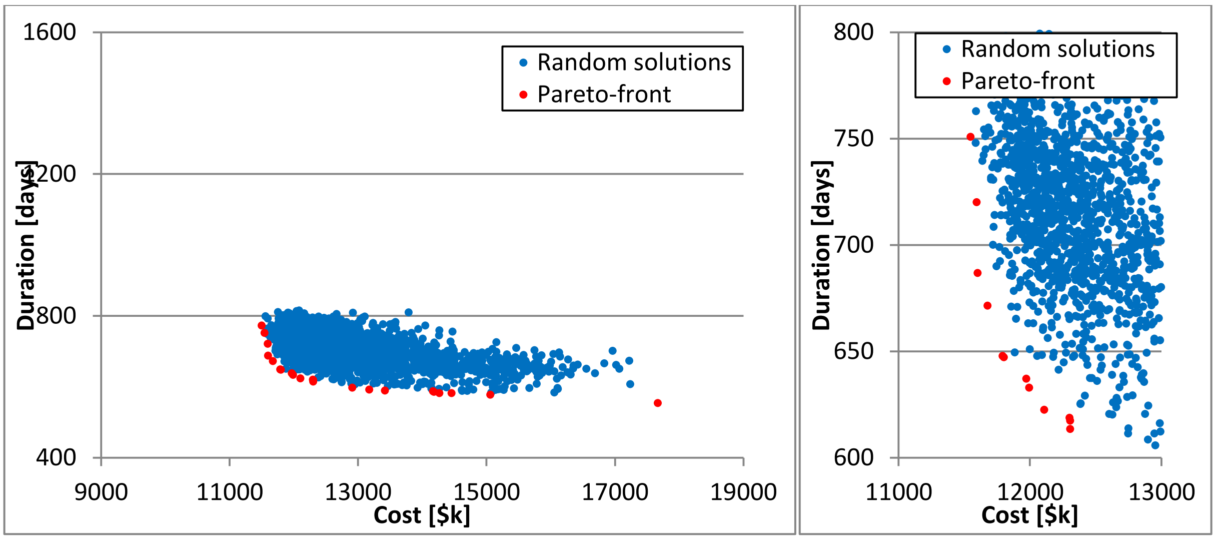
(c) Overall objective space (d) Pareto region

Figure 7.21: Pareto-front and random solutions for the hood process assuming work policy P_1.



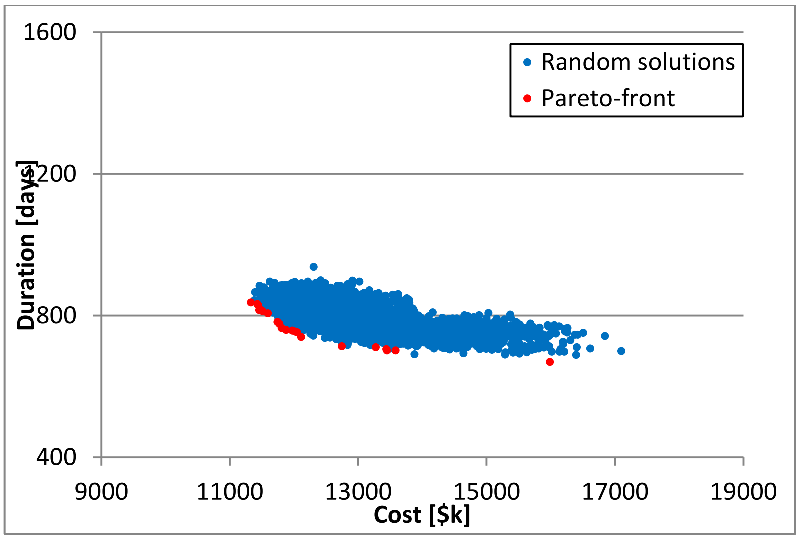
(c) Overall objective space (d) Pareto region

Figure 7.22: Pareto-front and random solutions for the hood process assuming work policy P_2.

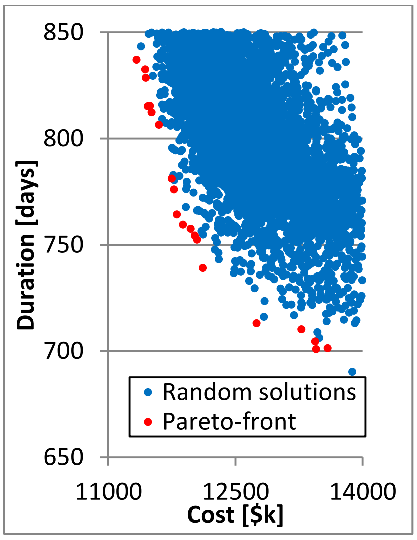


(c) Overall objective space (d) Pareto region

Figure 7.23: Pareto-front and random solutions for the hood process assuming work policy P_3

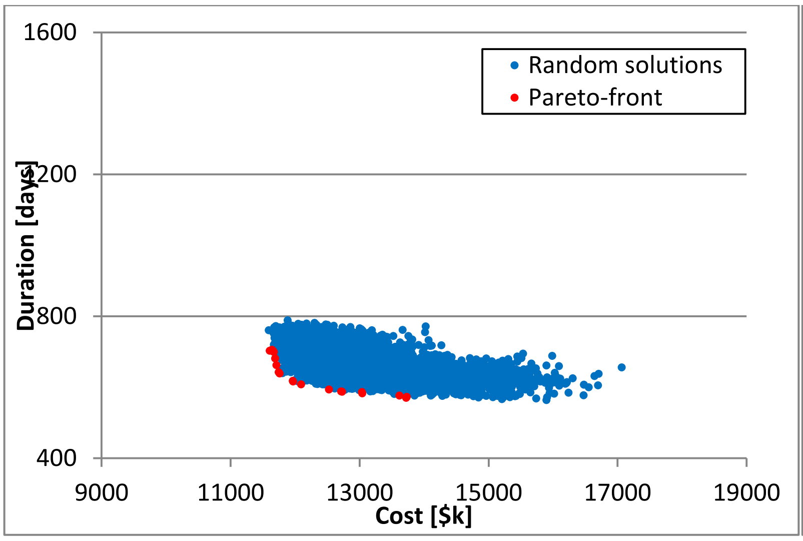


(e) Overall objective space

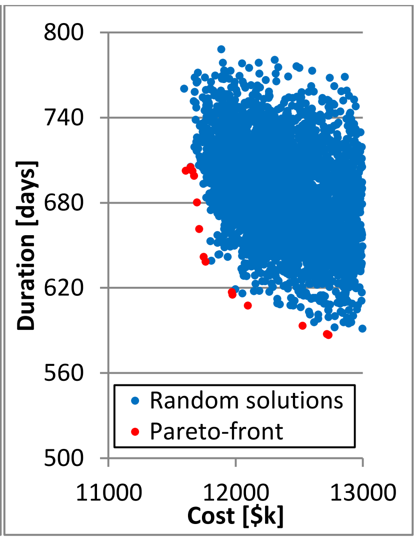


(f) Pareto region

Figure 7.24: Pareto-front and random solutions for the hood process assuming work policy P_4.



(e) Overall objective space



(f) Pareto region

Figure 7.25: Pareto-front and random solutions for the hood process assuming work policy P_5.

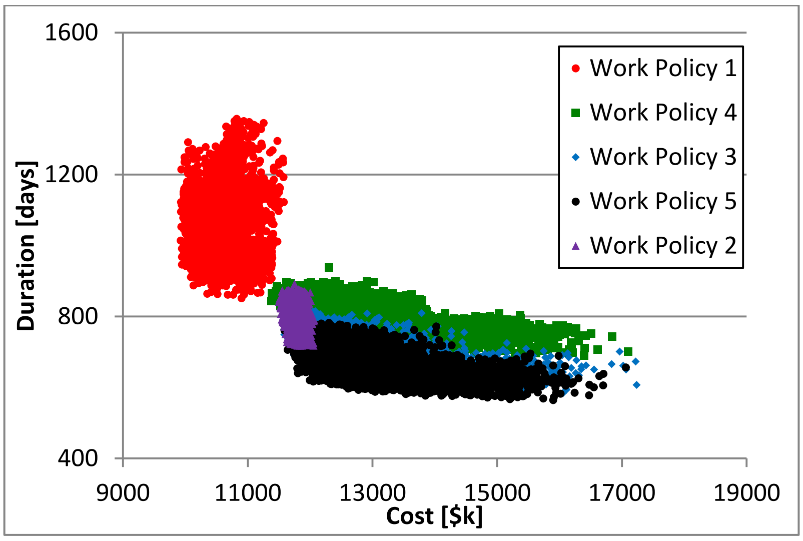


Figure 7.26: Comparison of random solutions for the hood process assuming work policies P_1-P_5.

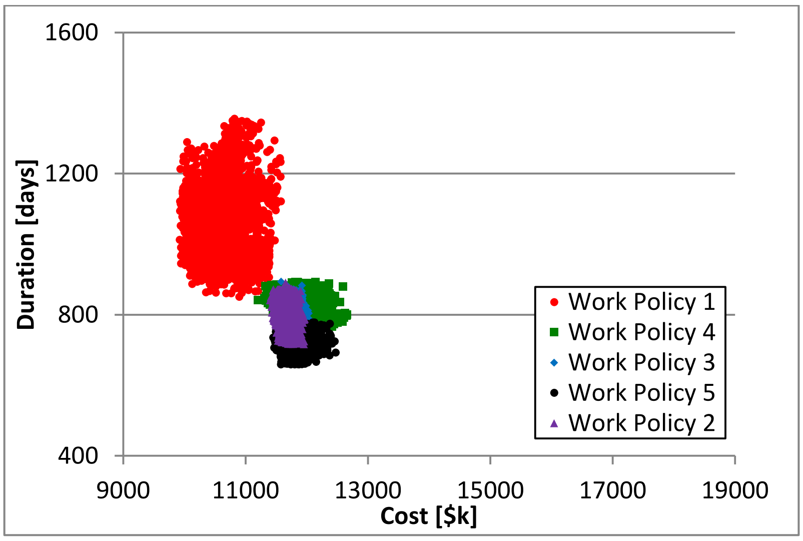


Figure 7.27: Comparison of random solutions for the hood process assuming work policies P_1-P_5 without crashing and overlapping.

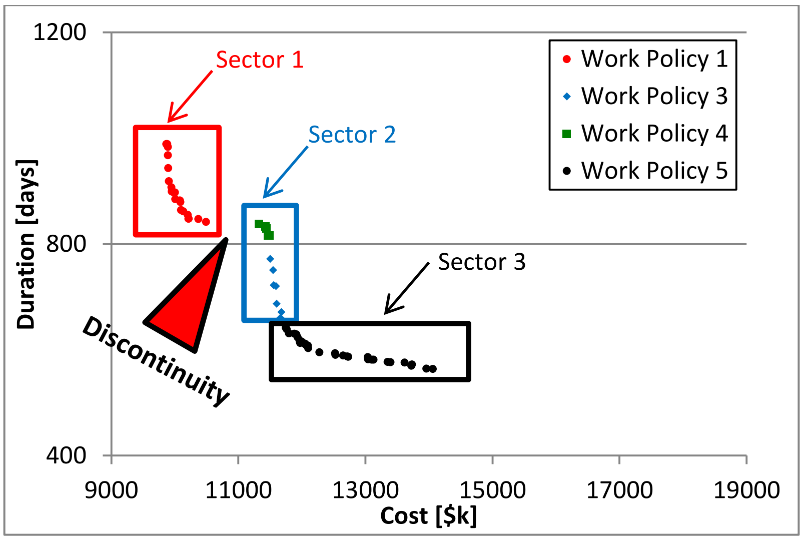


Figure 7.28: Overall Pareto-front for the hood process assuming work policies P_1-P_5.

the variance of process cost and duration due to changes of the activity sequence for different work policies: policy P_1 clearly features the greatest time-cost spreads whereas policies P_2-P5, which are all less restrictive with respect to the predecessor information needed, exhibit fewer variances.

The greatest, and probably most important, difference between the hood development process and the UCAV case study is the shape of the overall Pareto-front. While the UCAV process exhibited a continuous Pareto-front, we yielded a discontinuous one for the hood process. According to our analysis in chapter 5, such discontinuities in the Pareto-front are supposed to occur with more feedback relationships. Hence, the underlying case study verified this important insight. To examine the solutions on the overall Pareto-front more in detail, we decomposed it into three sectors: 1) sector 1 includes the cheapest but slowest processes exclusively generated by work policy P_1 2) Pareto-solutions produced by work policies P_3 and P_4 appear in the second sector while 3) sector 3 features the collectively fastest processes associated with highest cost. The evaluation of certain process parameters for each sector is proposed in Table 7.5 allowing the following conclusions:

1. The discontinuity with respect to cost in the Pareto-front between sectors 1 and 2 is not a result of different feedback densities or longer feedback cycles (distance to the diagonal). Rather, it is more likely caused by the work policy rule regarding the parallel execution of non-adjacent activities. Whereas P_2–P_5 allow the simultaneous execution of non-adjacent activities, P_1 does not. Thus, the processes in sectors 2 and 3 (obtained through the application of P_2-P_5) exhibited more parallelism within the process and consequently a higher number of costly reworks/partial reworks.
2. The actual amount of rework/partial rework may not be an immediate result of process architecture (e.g. number of reworks) but is also work policy driven.

Parameter	**Sector 1**	**Sector 2**	**Sector 3**
Dominating work policy	P_1	P_3	P_4
Average number of feedback relationships	65.4	63.8	77.7
Average distance of a feedback mark to diagonal	7.16	6.94	9.04
Weighted average distance of a feedback mark to diagonal	1.92	1.88	2.34
Average number reworks	59.3	155.3	158.9
Average number partial reworks	51.3	60.2	81.2
Average crashing intensity (1st iteration) for any activity (min: 0, max: 0.15)	0	0.07	0.08
Average crashing intensity (1st iteration) for critical activities 26, 27, 28 and 32 (min: 0, max: 0.21)	0	0.05	0.09
Average crashing intensity (1st iteration) for most critical activity 26 (min: 0, max: 0.25)	0	0.07	0.18
Average intensity for overlapping available (1st iteration) between two activities (min: 0.86, max: 1.0)	0	0.92	0.93
Average overlapping received intensity (1st iteration) between two activities (min: 0, max: 0.2)	0	0.09	0.11

Table 7.5: Selected process parameters for four disjoint sectors of the overall Pareto-front.

Justification of model complexity

At last, we were interested if changes of the Pareto-front due to the addition/reduction of model complexity are as evident as in case of the UCAV process and hence computed the Pareto-fronts for different levels of model complexity. Exemplarily displaying the corresponding Pareto-fronts for work policy P_3 (Figure 7.29) illuminates their differences and highlights once more the need for a complex process model. As opposed to the UCAV, however, process architecture is not the greatest lever for process cost & duration process. We suspected this outcome due to the high amount of firm predecessor constraints which simply prevent the generation of distinct process architectures. In lieu, modifications of the individual crashing intensities mainly affect the results for process lead-time and cost.

7.3 Summary

In the previous sections we applied our new process model, simulation and time-cost optimization approach to two PD processes in practice with varying complexity and industry affiliations. By means of these two practical examples we intended to highlight the shortcomings of past research thereby justifying the need for our study. Moreover, the managerial insights gained in chapter 5 were based on artificially constructed process and

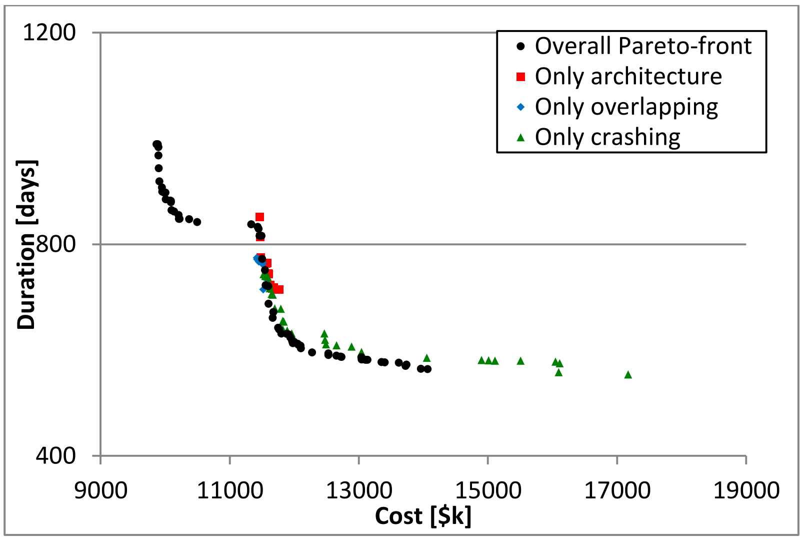


Figure 7.29: Comparison of the overall Pareto-front (all work policies) for the hood process with the Pareto-fronts assuming work policy P_3 for different levels of complexity.

should be hence verified with actual processes in practice.

In fact, the two case studies insistently demonstrated the need for a sophisticated joint approach of process modeling, simulation and optimization. Assuming different levels of model complexity, the Pareto-fronts for the two case study processes showed substantial differences. This observation let us conclude that a realistic time-cost estimation (which is the basis for a subsequent optimization) is only possible when considering all fundamental process parameters simultaneously. Besides, regarding optimization, we compared the obtained Pareto-optimal points with random solutions. Thereby, we detected substantial differences consequently pointing out the need for an intelligent optimization strategy instead of a random “guess”.

Referring to the verification of previous insights, we could actually confirm some of them, in particular the impact of work policy on process time &cost. First of all, we demonstrated that time-cost tradeoffs also emerge for the two practical processes due to the choice of work policy rules (and not only through overlapping- and crashing intensity or process architecture modifications). Moreover, the Pareto-fronts (and approximated objective spaces) exhibited a similar shape like the theoretically derived ones in chapter 5. Importantly, the time-cost advantages/disadvantages of the individual work policies were identical for case studies and theoretical processes. Besides, the global Pareto-front (including the consideration of all process parameters) of the second, more iterative, featured discontinuities – just like we pointed out for artificial processes. Last but not least we could validate our assumption in section 4.6 referring to the impact of changes in the activity sequence on the variance of time and cost: in fact, work policy P_1 provoked a higher spread in time & cost as a consequence of changes in the activity sequence than the other work policies.

Part IV
CONCLUSIONS

8. Conclusions and Future Work

8.1 Conclusions

The time-cost tradeoff problem for PD processes constitutes a problem of great practical relevance as PD processes may involve an enormous amount of (expensive) resources employed over a long period of time. Consequently, managers are anxious to minimize cost and duration simultaneously. Though, past research pointed out that cost reducing strategies extend process duration and vice versa time cutting methods boost process cost. Thus, a tradeoff problem arises making efficient process planning mandatory in order to avoid a waste of money and/or time. Since processes can be regarded as systems, the design of optimal processes is clearly associated with the (systems) engineering community. Indeed, the actual product is not engineered but the related development process.

In fact, the literature on the time-cost tradeoff problem extensively studied the effects of time- and cost reducing methods and even proposed optimization strategies to identify the set of all best tradeoff processes. But publications to date lacked in analyzing the impact of feedbacks on these time/cost minimizing strategies and on the time-cost solutions themselves – although feedbacks are likely to occur in PD processes and greatly affect time & cost. Besides, previous studies merely considered a subset of process parameters concurrently for the time-cost analysis. As a consequence, the insights and contributions of previous literature regarding practicability must be questioned. This research gap was the motivation for the underlying thesis: to help engineers developing better products because of faster and cheaper processes. For this purpose, we developed a new PD process model which accounts for feedbacks in the process and jointly considers the process parameters influencing time and cost most. Particularly notable is the introduction of several work policies. Furthermore, we proposed a time-cost simulation tailored to this model as well as an optimization strategy capable of detecting the Pareto-front solutions for a process.

We used the new model, simulation and optimization to derive numerous managerial insights. These insights are representative as they are based on millions of artificially constructed processes and (to some extent) verified by two real-world processes. In our opinion, the most important insights and conclusions are the following:

1. The scale-up behavior for cost and duration of cyclic processes mainly depends on learning and rework impact rather than number of relationships. In most cases, time and cost will increase logarithmically with the number of relationships in the process. Though, the scale-up becomes exponential if learning occurs at a constant rate and the number of maximal iterations is not limited. From a practical perspective, the logarithmical scale-up behaviors are interesting as they suggest that product quality, which is influenced by the number of iterations, could be strongly enhanced at a more or less moderate increase of overall cost and duration.
2. The introduced work policies cause different outcomes for process cost and duration. Interestingly, no work policy appeared to be superior to the others in both (cost, duration) dimensions. Instead, a time-cost tradeoff due to the choice of work policy arises. Therefore we suggest extending the time-cost tradeoff problem by a further managerial lever (beside crashing, overlapping and process architecture), namely work policy. The substantial time-cost differences for the work policies should also sensitize engineers/managers for the impact of this managerial instrument.
3. With increasing dynamics, the application of crashing and overlapping becomes more

economic. Hence, the application of these two Concurrent Engineering strategies is highly favorable for dynamic process environments. As for crashing, the type of resources used for crashing is crucial. In general, managers should use resources which have to be paid only once for all potential iterations of an activity, e.g. machinery or software. With respect to overlapping, we found that an extended critical path length (due to rework) amplifies the temporal benefits of overlapping two activities. This effect may outweigh the percental cost increase associated with overlapping, which exhibits a linear growth. Interestingly, overlapping slightly reduces the number of reworks, but on the other hand causes an exponential enlargement of partial reworks which make the process significantly more expensive.

4. The primary type of information flow within a cyclic process affects economic efficiency of crashing and overlapping. In average, crashing and overlapping become more economic with increasing number of parallel activities, i.e. activities which are uncoupled with respect to feed-forward information flow. Contrary, processes featuring a strictly sequential information flow performed worst.
5. The time-cost optimization of real-world processes clearly highlighted the need for a sophisticated process model. Assuming different levels of model complexity, the corresponding Pareto-fronts for the two case study processes showed substantial differences. This observation let us conclude that only the combined consideration of all fundamental process parameters for modeling, simulation and subsequent optimization is pivotal to gain “global” Pareto-optimal processes.

8.2 Future work

In consequence of the practical relevance of the time-cost tradeoff problem in PD and the results of our work, we encourage scientists to continue research on this problem. Some of the possible directions for future research are the following ones:

1. One of the weaknesses of the proposed model is the missing consideration of resource constraints. Resources like staff, machinery or facilities – just to name a few – must be assigned to PD activities and are typically limited. Therefore, activities can be only executed if the necessary resources are available. Hence, we suggest to extend the model by resource constraints and to study their impact on the scale-up behavior for time & cost as well as on the Pareto-front.
2. Beside cost and duration, technical performance/quality is another dimension to be considered for process planners. Therefore, we encourage the extension of our process model/simulation/optimization by measurements for technical performance of the product to be developed. Obviously, quality is expected to increase with growing number of iterations in the process and we accounted for iterations. Thus, we indirectly considered quality issues. Though, we did not explicitly incorporate quality related model parameters.
3. We strongly recommend further research on closed form analysis for the approximation of time and cost for cyclic processes. A realistic proxy could be used as a deterministic objective function for the multi-objective optimization and is essential for very large problem sets involving several hundred or thousand activities. Otherwise, using simulation as objective functions for large problem sets would require a high computational effort. Indeed, research towards closed-form solutions is very challenging and we cannot say if it will succeed for arbitrary processes. Though, the potential benefits of a closed-form are worth trying.

Appendix

A: Simulation results for section 5.1.3

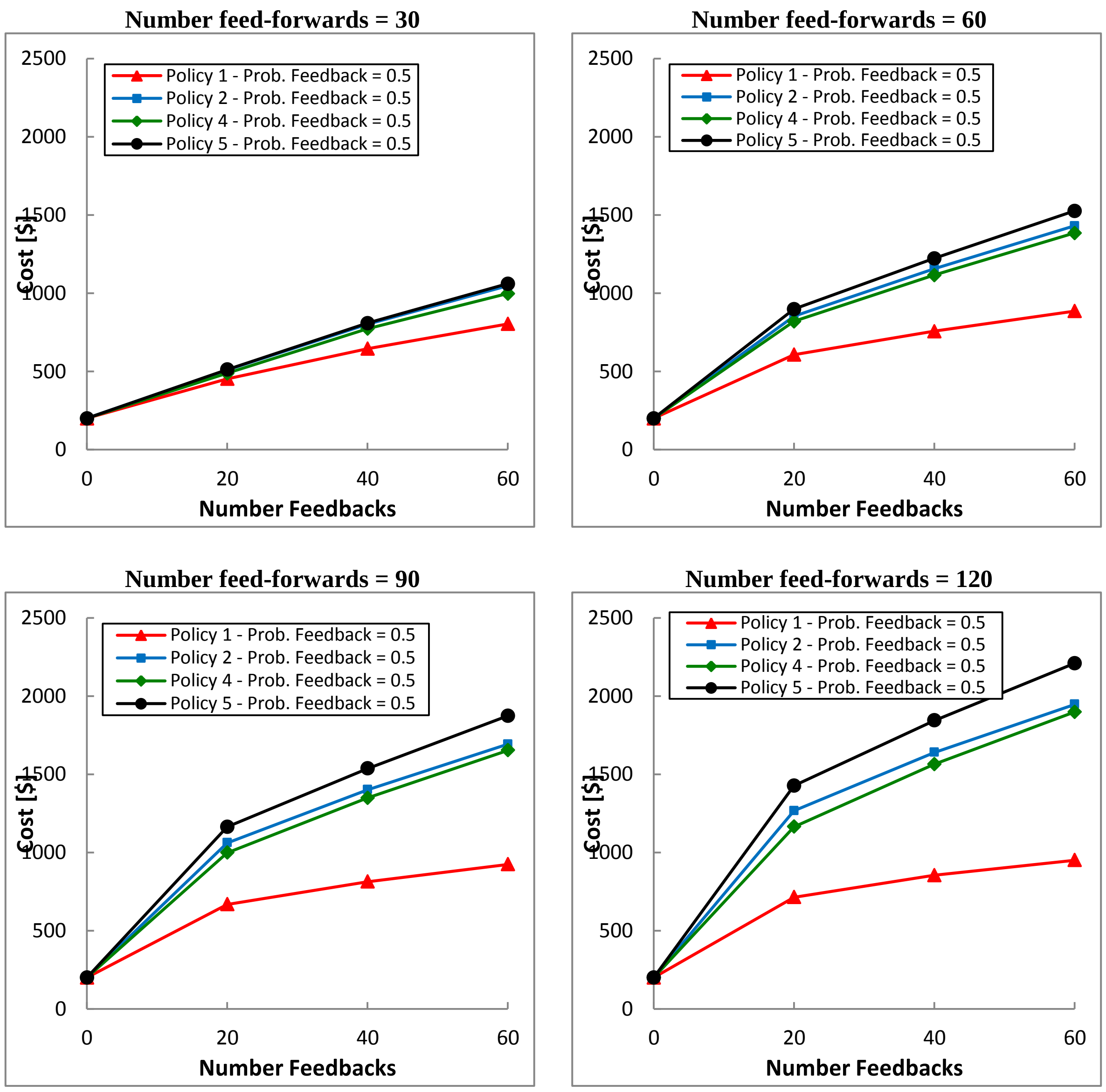


Figure A.1: Comparison of cost outcomes for work policies P_1, P_2, P_4 and P_5 assuming sequential processes and a feedback probability 0.5.

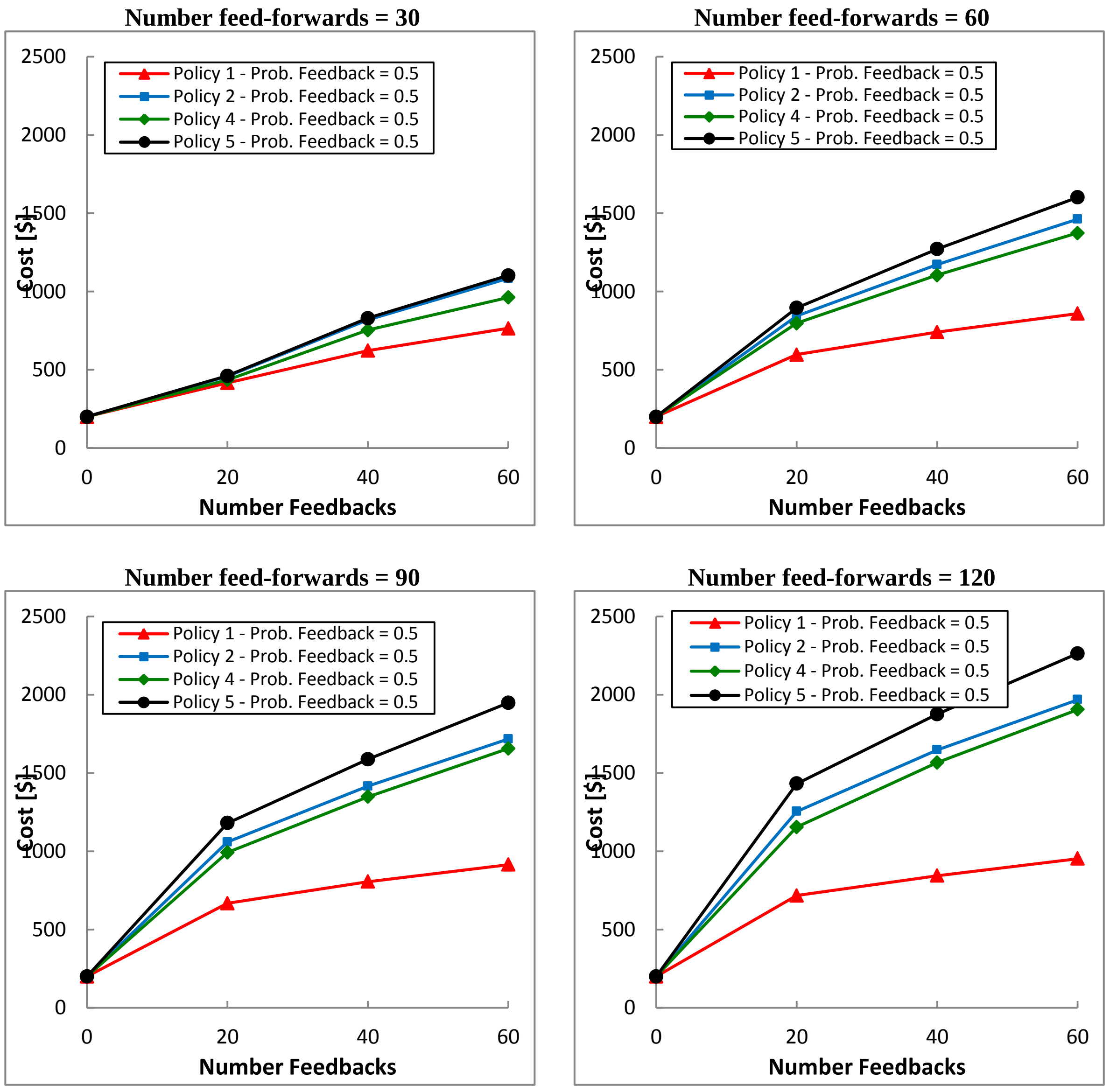


Figure A.2: Comparison of cost outcomes for work policies P_1, P_2, P_4 and P_5 assuming mixed processes and a feedback probability 0.5.

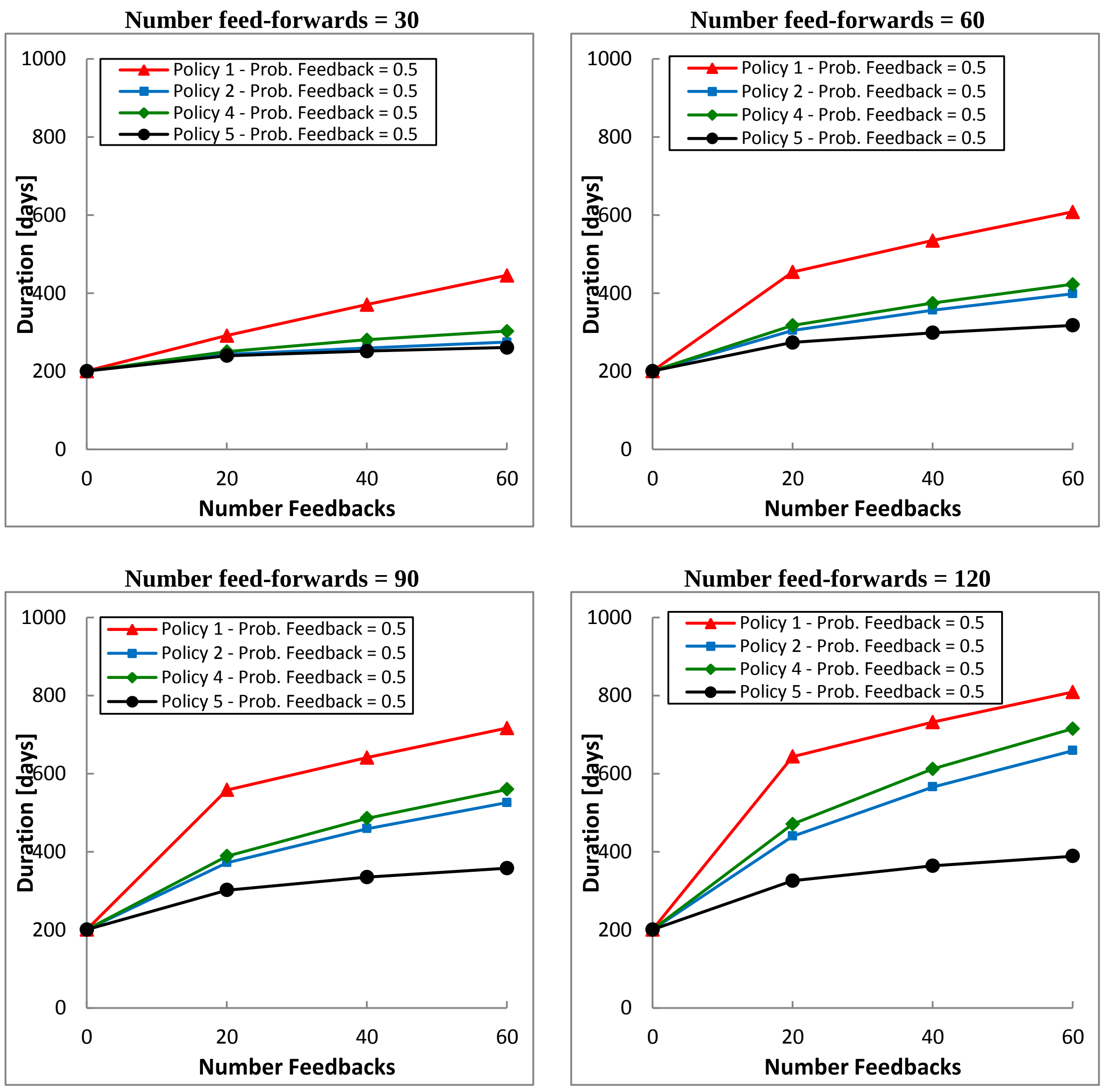


Figure A.3: Comparison of schedule outcomes for work policies P_1, P_2, P_4 and P_5 assuming sequential processes and a feedback probability 0.5.

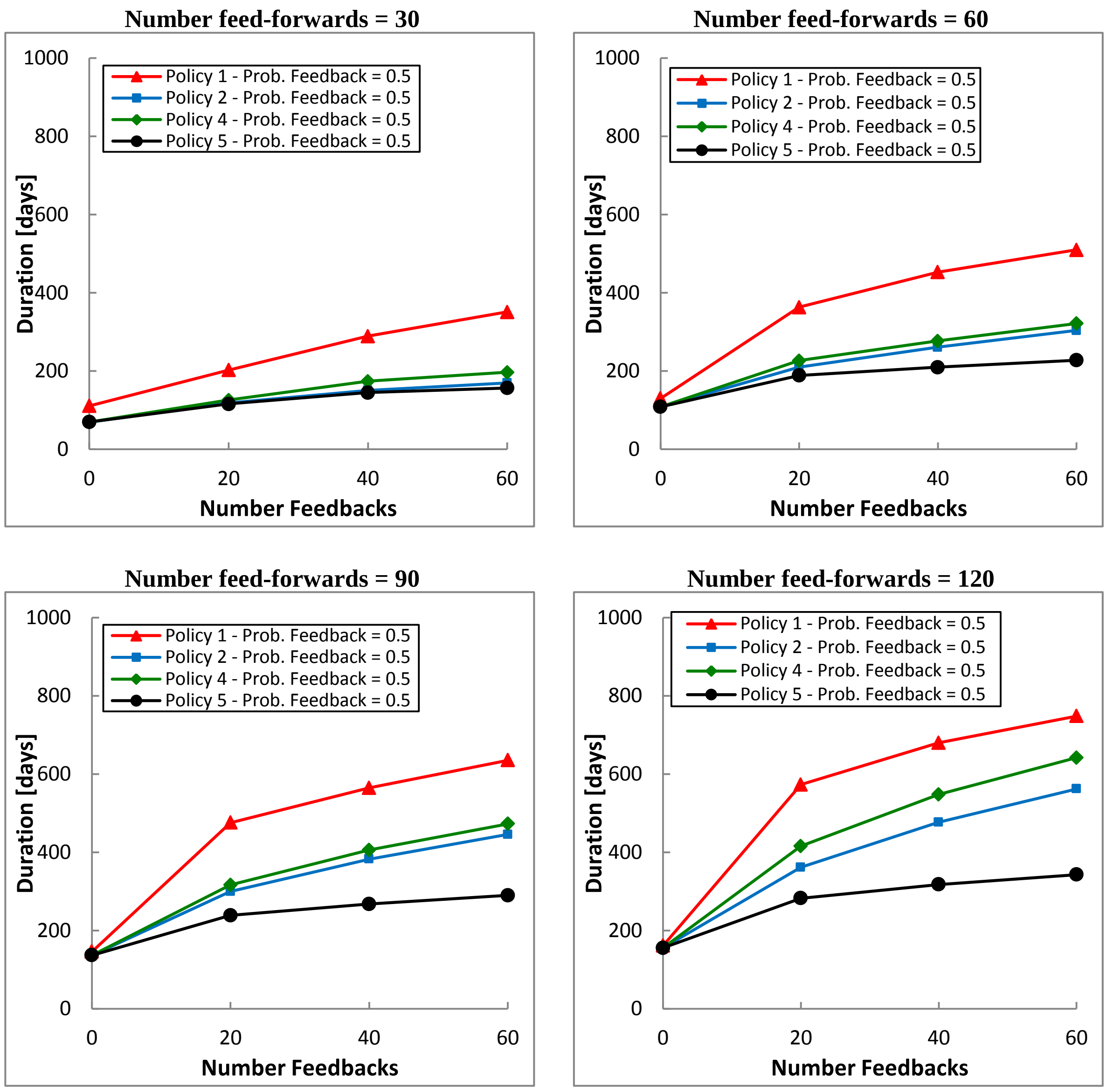


Figure A.4: Comparison of schedule outcomes for work policies P_1, P_2, P_4 and P_5 assuming mixed processes and a feedback probability 0.5.

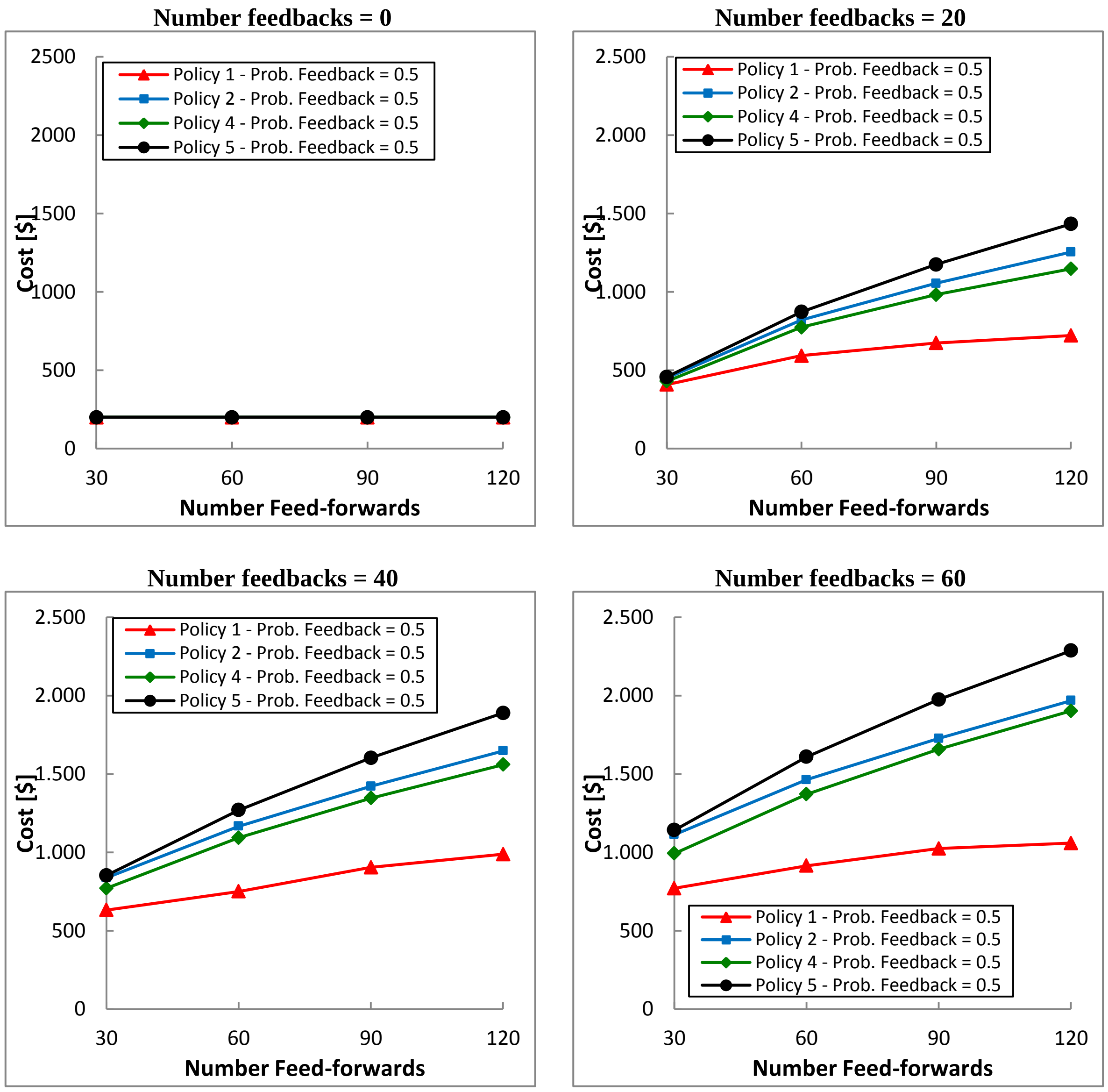


Figure A.5: Comparison of cost outcomes for work policies P_1, P_2, P_4 and P_5 assuming random processes and different number of feedbacks with feedback probability 0.5.

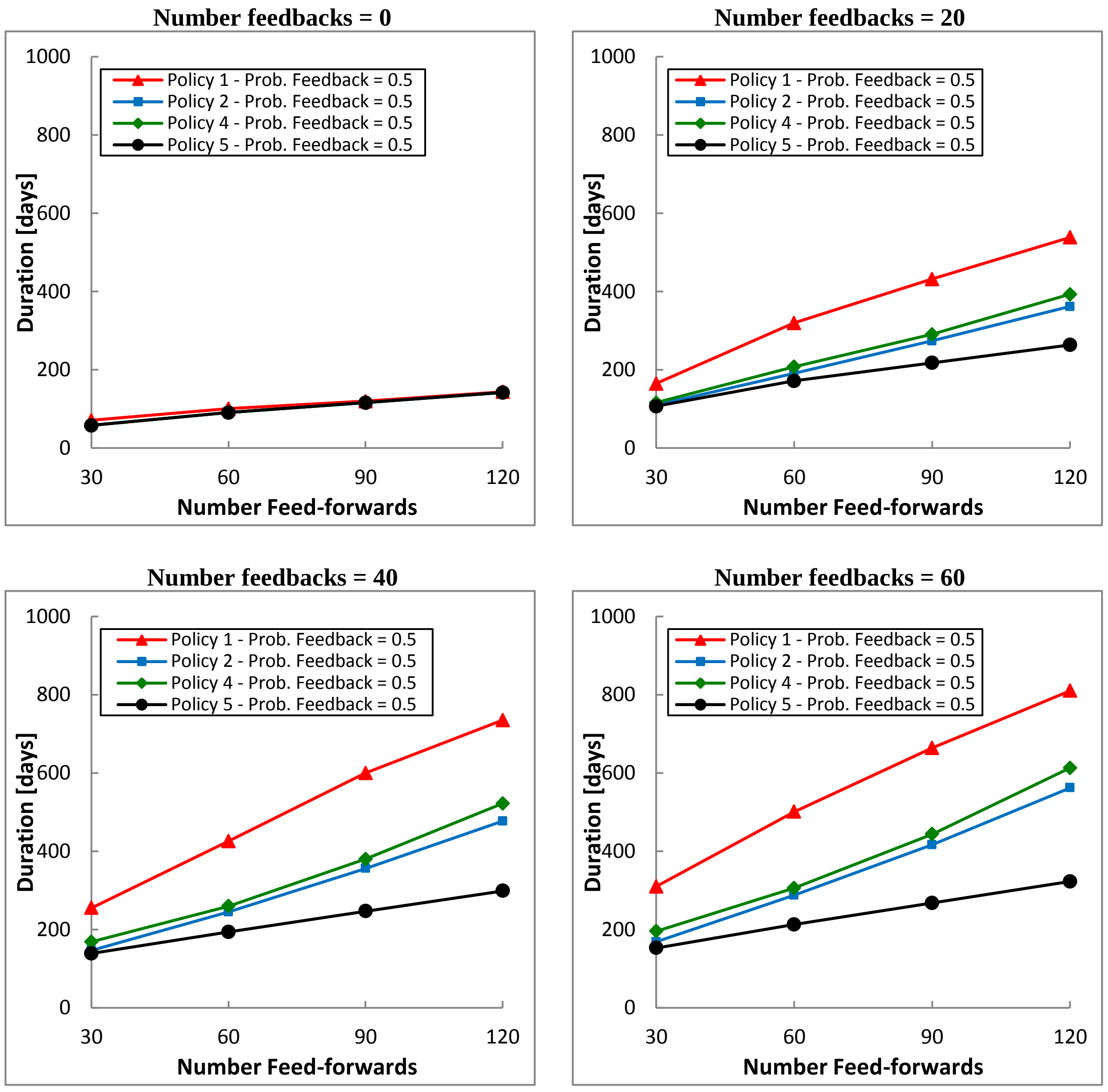


Figure A.6: Comparison of schedule outcomes for work policies P_1, P_2, P_4 and P_5 assuming random processes and different number of feedbacks with feedback probability 0.5.

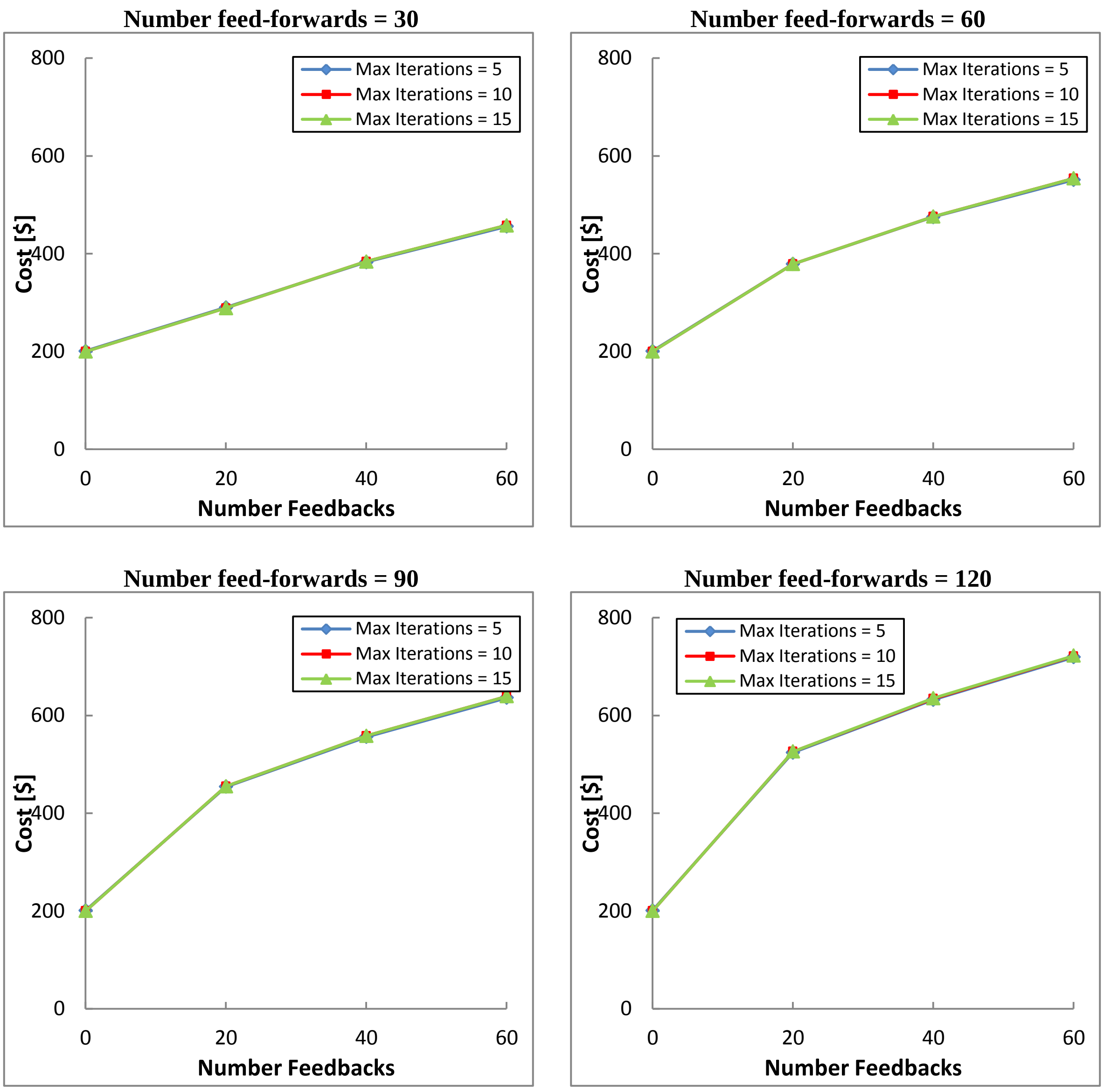


Figure A.7: Comparison of cost outcomes for different maximal number of iterations allowed assuming P_1, random processes and a feedback probability 0.5.

B: Simulation results for section 5.1.4

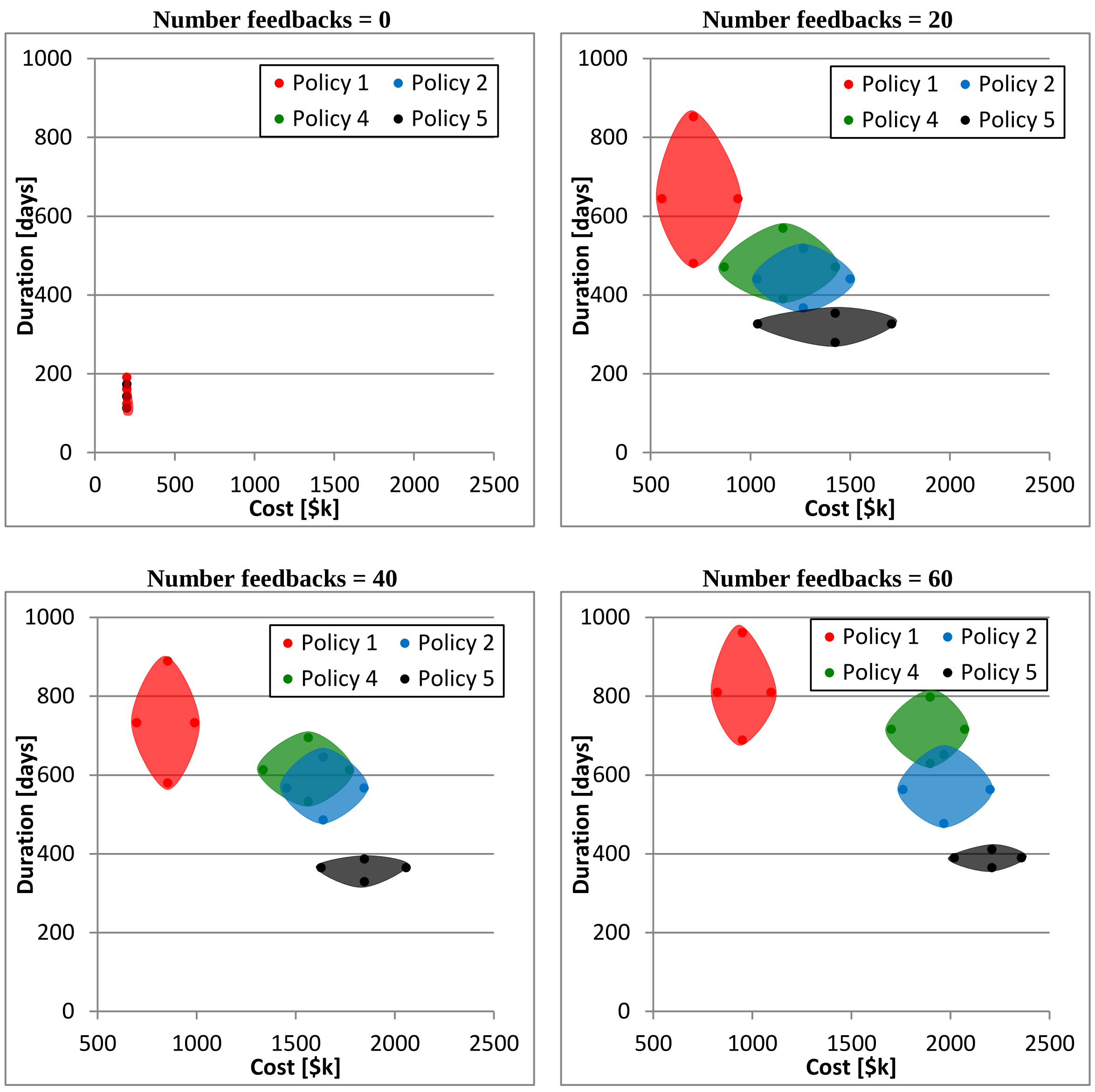


Figure B.8: Time-Cost approximation of objective space for policies P_1, P_2, P_4 and P_5 assuming sequential processes with 120 feed-forwards and 0.5 feedback probability.

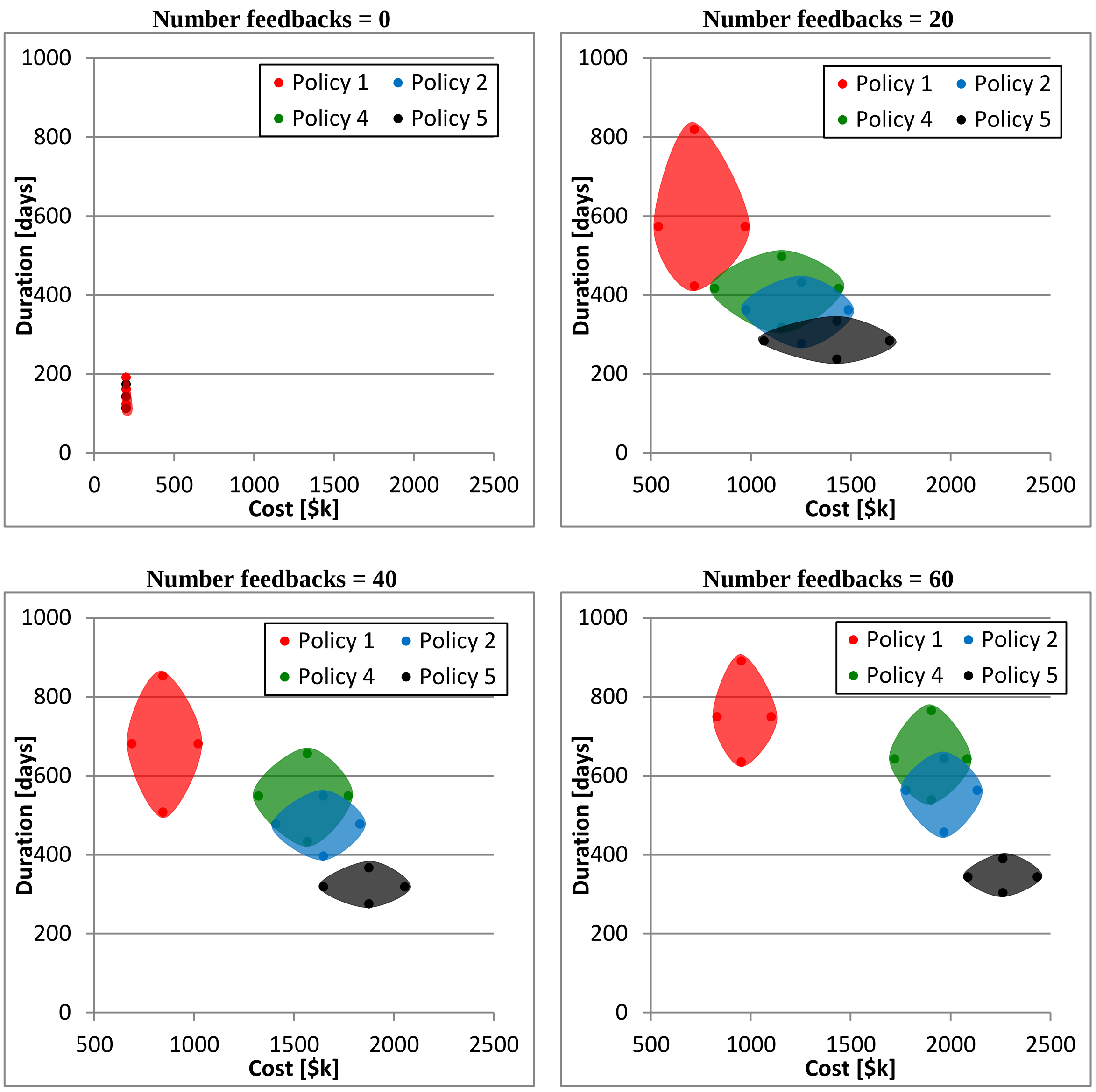


Figure B.9: Time-Cost approximation of objective space for policies P_1, P_2, P_4 and P_5 assuming mixed processes with 120 feed-forwards and 0.5 feedback probability.

C: Simulation results for section 5.2.3

		Random processes – Multi Payment Crashing											
		Feedback probability 0.1				**Feedback probability 0.3**				**Feedback probability 0.5**			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
	60	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
	90	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
	120	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
Average		-1				-1				-1			
Average		-1											

Table C.1: ϕ values for the base case of P_{3a} assuming random processes.

		Sequential processes – Multi Payment Crashing											
		Feedback probability 0.1				**Feedback probability 0.3**				**Feedback probability 0.5**			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
	60	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
	90	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
	120	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
Average		-1				-1				-1			
Average		-1											

Table C.2: ϕ values for the base case of P_{3a} assuming sequential processes.

		Mixed processes – Multi Payment Crashing											
		Feedback probability 0.1				**Feedback probability 0.3**				**Feedback probability 0.5**			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
	60	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
	90	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
	120	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
Average		-1				-1				-1			
Average		-1											

Table C.3: ϕ values for the base case of P_{3a} assuming mixed processes.

		Random processes – Single Payment Crashing											
		Feedback probability 0.1				**Feedback probability 0.3**				**Feedback probability 0.5**			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	-1	-1	-1.01	-1.01	-1	-1.02	-1.05	-1.08	-1	-1.07	-1.15	-1.20
	60	-1	-1	-1.01	-1.01	-1	-1.03	-1.10	-1.14	-1	-1.14	-1.19	-1.21
	90	-1	-1	-1.01	-1.01	-1	-1.07	-1.14	-1.16	-1	-1.18	-1.20	-1.21
	120	-1	-1	-1	-1.01	-1	-1.12	-1.16	-1.17	-1	-1.20	-1.21	-1.21
Average		-1.004				-1.078				-1.136			
Average		-1.073											

Table C.4: ϕ values for the base case of P_{3a} assuming random processes.

		Sequential processes – Single Payment Crashing											
		Feedback probability 0.1				**Feedback probability 0.3**				**Feedback probability 0.5**			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	-1	-1	-1	-1	-1	-1	-1.02	-1.05	-1	-1.05	-1.09	-1.13
	60	-1	-1	-1	-1	-1	-1.04	-1.09	-1.12	-1	-1.13	-1.15	-1.17
	90	-1	-1	-1	-1	-1	-1.07	-1.13	-1.14	-1	-1.16	-1.17	-1.18
	120	-1	-1	-1	-1.01	-1	-1.12	-1.15	-1.16	-1	-1.19	-1.19	-1.19
Average		-1				-1.068				-1.113			
Average		-1.06											

Table C.5: ϕ values for the base case of P_{3a} assuming sequential processes.

		Mixed processes – Single Payment Crashing											
		Feedback probability 0.1				**Feedback probability 0.3**				**Feedback probability 0.5**			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	-1	-1	-1.01	-1.01	-1	-1.02	-1.03	-1.07	-1	-1.06	-1.14	-1.19
	60	-1	-1	-1	-1.01	-1	-1.04	-1.10	-1.14	-1	-1.15	-1.19	-1.21
	90	-1	-1	-1	-1.01	-1	-1.08	-1.14	-1.16	-1	-1.18	-1.19	-1.20
	120	-1	-1	-1	-1.01	-1	-1.12	-1.16	-1.16	-1	-1.20	-1.20	-1.20
Average		-1.003				-1.076				-1.132			
Average		-1.07											

Table C.6: ϕ values for the base case of P_{3a} assuming mixed processes.

		Random processes – Once Payment Crashing											
		Feedback probability 0.1				**Feedback probability 0.3**				**Feedback probability 0.5**			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	-1	-1.05	-1.11	-1.18	-1	-1.28	-1.94	-3.41	-1	-2.24	-4.15	-5.54
	60	-1	-1.07	-1.15	-1.25	-1	-1.83	-4.45	-6.45	-1	-4.05	-5.77	-7.20
	90	-1	-1.08	-1.19	-1.33	-1	-3.07	-6.22	-7.69	-1	-5.20	-7	-8.49
	120	-1	-1.10	-1.23	-1.43	-1	-4.41	-7.26	-8.85	-1	-6.16	-8.08	-9.63
Average		-1.136				-3.804				-4.844			
Average		-3.261											

Table C.7: ϕ values for the base case of P_{3a} assuming random processes.

		Sequential processes – Once Payment Crashing											
		Feedback probability 0.1				**Feedback probability 0.3**				**Feedback probability 0.5**			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	-1	-1.06	-1.13	-1.22	-1	-1.36	-2.04	-3.15	-1	-2.54	-4	-5.2
	60	-1	-1.07	-1.16	-1.27	-1	-2.01	-4.67	-6.27	-1	-4.24	-5.74	-7.08
	90	-1	-1.08	-1.19	-1.34	-1	-3.44	-6.19	-7.61	-1	-5.25	-6.91	-8.32
	120	-1	-1.10	-1.23	-1.44	-1	-4.70	-7.25	-8.82	-1	-6.25	-8.08	-9.51
Average		-1.143				-3.844				-4.82			
Average		-3.269											

Table C.8: ϕ values for the base case of P_{3a} assuming sequential processes.

		Mixed processes – Once Payment Crashing											
		Feedback probability 0.1				**Feedback probability 0.3**				**Feedback probability 0.5**			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	-1	-1.05	-1.11	-1.18	-1	-1.29	-1.90	-3.11	-1	-2.27	-4.07	-5.39
	60	-1	-1.07	-1.15	-1.25	-1	-1.88	-4.60	-6.40	-1	-4.19	-5.80	-7.20
	90	-1	-1.08	-1.19	-1.33	-1	-3.27	-6.16	-7.64	-1	-5.23	-6.97	-8.42
	120	-1	-1.10	-1.23	-1.43	-1	-4.62	-7.33	-8.94	-1	-6.18	-8.10	-9.61
Average		-1.136				-3.821				-4.839			
Average		-3.265											

Table C.9: ϕ values for the base case of P_{3a} assuming mixed processes.

		Random processes – Single Payment Crashing											
		Feedback probability 0.1				**Feedback probability 0.3**				**Feedback probability 0.5**			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	20	19.9	19.8	19.7	20	19.6	19.1	18.5	20	18.8	17.5	16.7
	60	20	19.9	19.9	19.9	20	19.4	18.2	17.5	20	17.5	16.9	16.6
	90	20	19.9	19.9	19.8	20	18.6	17.6	17.3	20	17.0	16.8	16.6
	120	20	20	19.9	19.9	20	17.9	17.3	17.2	20	16.7	16.7	16.6
Average		19.91				18.64				17.78			
Average		18.77											

Table C.10: $\Delta c_{c_{tot}/c}$ values for the base case assuming random processes.

		Sequential processes – Single Payment Crashing											
		Feedback probability 0.1				**Feedback probability 0.3**				**Feedback probability 0.5**			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	20	20	20	20	20	20	19.7	19.1	20	19.1	18.3	17.7
	60	20	20	20	20	20	19.4	18.4	17.8	20	17.8	17.4	17.2
	90	20	20	20	19.9	20	18.6	17.8	17.6	20	17.3	17.1	17.1
	120	20	20	20	19.9	20	17.9	17.5	17.4	20	16.9	17	17
Average		19.99				18.83				18.12			
Average		18.98											

Table C.11: $\Delta c_{c_{tot}/c}$ values for the base case assuming sequential processes.

		Mixed processes – Single Payment Crashing											
		Feedback probability 0.1				**Feedback probability 0.3**				**Feedback probability 0.5**			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	20	19.9	19.9	19.8	20	19.7	19.3	18.6	20	18.8	17.6	16.8
	60	20	19.9	19.9	19.9	20	19.3	18.3	17.6	20	17.5	16.9	16.6
	90	20	20	19.9	19.9	20	18.5	17.6	17.4	20	17.1	16.9	16.7
	120	20	20	19.9	19.9	20	17.9	17.4	17.3	20	16.8	16.8	16.8
Average		19.93				18.68				17.83			
Average		18.81											

Table C.12: $\Delta c_{c_{tot}/c}$ values for the base case assuming mixed processes.

		Random processes – One-time Payment Crashing											
		Feedback probability 0.1				**Feedback probability 0.3**				**Feedback probability 0.5**			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	20	19	17.9	16.9	20	15.6	10.3	5.9	20	8.9	4.8	3.6
	60	20	18.7	17.4	16.1	20	10.9	4.5	3.1	20	4.9	3.5	2.8
	90	20	18.5	16.9	15.1	20	6.5	3.2	2.6	20	3.9	2.9	2.4
	120	20	18.2	16.3	14	20	4.5	2.8	2.3	20	3.3	2.5	2.1
Average		17.813				9.513				7.85			
Average		11.725											

Table C.13: $\Delta c_{c_{tot}/c}$ values for the base case of P_{3a} assuming random processes.

		Sequential processes – One-time Payment Crashing											
		Feedback probability 0.1				**Feedback probability 0.3**				**Feedback probability 0.5**			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	20	18.8	17.6	16.4	20	14.7	9.8	6.4	20	7.9	5	3.8
	60	20	18.6	17.2	15.8	20	9.9	4.3	3.1	20	4.7	3.5	2.8
	90	20	18.5	16.7	14.9	20	5.8	3.2	2.6	20	3.8	2.9	2.4
	120	20	18.2	16.2	13.9	20	4.3	2.8	2.3	20	3.2	2.5	2.1
Average		17.675				9.325				7.787			
Average		11.596											

Table C.14: $\Delta c_{c_{tot}/c}$ values for the base case of P_{3a} assuming sequential processes.

		Mixed processes – One-time Payment Crashing											
		Feedback probability 0.1				**Feedback probability 0.3**				**Feedback probability 0.5**			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	20	19	18	17	20	15.5	10.5	6.4	20	8.8	4.9	3.7
	60	20	18.7	17.4	16	20	10.6	4.4	3.1	20	4.8	3.5	2.8
	90	20	18.5	16.8	15.1	20	6.1	3.2	2.6	20	3.8	2.9	2.4
	120	20	18.2	16.3	13.9	20	4.3	2.8	2.3	20	3.3	2.5	2.1
Average		17.806				9.488				7.844			
Average		11.713											

Table C.15: $\Delta c_{c_{tot}/c}$ values for the base case of P_{3a} assuming mixed processes.

		Random processes – Single Payment Crashing											
		Feedback probability 0.1				Feedback probability 0.3				Feedback probability 0.5			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	0	1.4	3.6	6.1	0	10.8	42.5	132	0	70.7	120	305
	60	0	2	5.2	9.1	0	37.5	219	377	0	198	248	461
	90	0	2.7	6.8	12.4	0	114	354	491	0	276	438	568
	120	0	3.2	8.6	16.8	0	202	438	591	0	341	520	663
Average		4.869				188.05				263			
Average		151.98											

Table C.16: Number of reworked activities for the base case of P_{3a} assuming random processes.

		Sequential processes – Single Payment Crashing											
		Feedback probability 0.1				Feedback probability 0.3				Feedback probability 0.5			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	0	2	5	8.2	0	14.9	50	117	0	97.3	214	309
	60	0	2.2	6	10.3	0	50.8	240	376	0	224	360	478
	90	0	3	7.2	13.1	0	137	361	496	0	294	451	581
	120	0	3.3	8.9	17.4	0	220	446	597	0	356	533	659
Average		5.413				194.11				284.77			
Average		161.43											

Table C.17: Number of reworked activities for the base case of P_{3a} assuming sequential processes.

		Mixed processes – Single Payment Crashing											
		Feedback probability 0.1				Feedback probability 0.3				Feedback probability 0.5			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	0	1.4	3.6	6.1	0	11	40	111	0	73.3	203	290
	60	0	2.1	5.3	9.3	0	40.9	223	376	0	209	346	463
	90	0	2.8	6.9	12.6	0	124	357	492	0	284	442	572
	120	0	3.3	8.7	17	0	208	441	594	0	346	526	668
Average		4.944				188.62				276.39			
Average		156.65											

Table C.18: Number of reworked activities for the base case of P_{3a} assuming mixed processes.

		Random processes – Single Payment Crashing											
		Feedback probability 0.1				Feedback probability 0.3				Feedback probability 0.5			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	0	0	0.1	0.7	0	1.5	9.3	45.2	0	20.9	120	248
	60	0	0	0	0.3	0	7.7	89.2	209	0	125	248	370
	90	0	0	0	0.4	0	47.7	209	301	0	231	348	454
	120	0	0	0	0.9	0	128	294	378	0	343	453	553
Average		0.15				107.48				219.62			
Average		109.08											

Table C.19: Number of partial reworks for the base case of P_{3a} assuming random processes.

		Sequential processes – Single Payment Crashing											
		Feedback probability 0.1				Feedback probability 0.3				Feedback probability 0.5			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	0	0	0	0	0	0	5.7	28.1	0	19.7	77.3	152
	60	0	0	0	0	0	9.2	88.6	176	0	115	189	268
	90	0	0	0	0	0	56.3	187	256	0	207	290	367
	120	0	0	0	0.4	0	137	276	347	0	327	408	482
Average		0.03				97.93				181.38			
Average		93.11											

Table C.20: Number of partial reworks for the base case of P_{3a} assuming sequential processes.

		Mixed processes – Single Payment Crashing											
		Feedback probability 0.1				Feedback probability 0.3				Feedback probability 0.5			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	0	0	0	0.4	0	1.2	8.1	36.6	0	20.6	112	236
	60	0	0	0	0.1	0	8.2	90.5	207	0	130	244	358
	90	0	0	0	0.1	0	52.1	203	289	0	226	332	431
	120	0	0	0	0.8	0	132	289	370	0	338	440	530
Average		0.09				105.42				212.35			
Average		105.95											

Table C.21: Number of partial reworks for the base case of P_{3a} assuming mixed processes.

		Random processes – Single Payment															
		α = 2				**α = 4**				**α = 8**				**α = 16**			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	-35.5	-35.9	-40	-41.5	-13.9	-14.8	-15.9	-16.4	-4.03	-4.28	-4.59	-4.82	-0.58	-0.62	-0.66	-0.69
	60	-35.5	-37.9	-38.7	-40.4	-13.9	-15.5	-16.1	-16.3	-4.03	-4.57	-4.78	-4.86	-0.58	-0.66	-0.69	-0.70
	90	-35.5	-37.9	-39	-39.9	-13.9	-15.9	-16.1	-16.3	-4.03	-4.72	-4.80	-4.85	-0.58	-0.68	-0.70	-0.70
	120	-35.5	-39.9	-38.8	-39	-13.9	-16	-16.1	**-16.1**	-4.03	-4.80	-4.82	-4.84	-0.58	-0.70	-0.70	-0.70
Sum		-38.18				-15.44				-4.55				-0.658			
Sum		-14.709															

Table C.22: ϕ values for random processes assuming different α values.

		Sequential processes – Single Payment															
		α = 2				**α = 4**				**α = 8**				**α = 16**			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	-35.5	-35.9	-37.2	-40.1	-13.9	-14.2	-15.2	-15.7	-4.03	-4.18	-4.39	-4.54	-0.58	-0.61	-0.63	-0.65
	60	-35.5	-37.8	-39.4	-39.3	-13.9	-15.3	-15.7	-15.9	-4.03	-4.51	-4.61	-4.67	-0.58	-0.65	-0.67	-0.68
	90	-35.5	-37.8	-39.5	-39.2	-13.9	-15.8	-15.8	-15.9	-4.03	-4.66	-4.70	-4.72	-0.58	-0.67	-0.68	-0.68
	120	-35.5	-38.5	-39.1	-39	-13.9	-16	-15.9	-15.8	-4.03	-4.77	-4.75	-4.74	-0.58	-0.69	-0.69	-0.69
Sum		-37.8				-15.18				-4.46				-0.644			
Sum		-14.520															

Table C.23: ϕ values for sequential processes assuming different α values.

		Mixed processes – Single Payment															
		α = 2				**α = 4**				**α = 8**				**α = 16**			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	-35.5	-37	-40.6	-41.6	-13.9	-14.5	-15.7	-16.4	-4.03	-4.27	-4.58	-4.81	-0.58	-0.61	-0.66	-0.69
	60	-35.5	-37.9	-38.6	-40	-13.9	-15.5	-16.1	-16.3	-4.03	-4.57	-4.75	-4.85	-0.58	-0.66	-0.69	-0.70
	90	-35.5	-37.9	-39	-39.5	-13.9	-15.9	-16.1	-16.3	-4.03	-4.72	-4.76	-4.80	-0.58	-0.67	-0.67	-0.69
	120	-35.5	-39.0	-38.8	-39	-13.9	-16	-16.1	**-16.1**	-4.03	-4.78	-4.77	-4.79	-0.58	-0.68	-0.68	-0.67
Sum		-38.18				-15.41				-4.54				-0.649			
Sum		-14.694															

Table C.24: ϕ values for mixed processes assuming different α values.

		Random processes – Once Payment															
		α = 2				**α = 4**				**α = 8**				**α = 16**			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	-35.5	-76.6	-134	-176	-13.9	-31.3	-57.6	-73.6	-4.03	-8.81	-16.7	-22	-0.58	-1.30	-2.42	-3.21
	60	-35.5	-105	-1.51	-187	-13.9	-49.2	-71.6	-87.1	-4.03	-16	-22.6	-27.8	-0.58	-2.35	-3.36	-4.20
	90	-35.5	-138	-165	-183	-13.9	-60.1	-79.1	-93.9	-4.03	-20.1	-27.1	-32.4	-0.58	-3.03	-4.08	-4.96
	120	-35.5	-144	-160	-186	-13.9	-68.2	-91.3	**-104**	-4.03	-23.8	-30.8	-36.1	-0.58	-3.61	-4.73	-5.63
Sum		-112.4				-57.66				-18.77				-2.83			
Sum		-47.91															

Table C.25: ϕ values for random processes assuming different α values.

		Sequential processes – Once Payment															
		α = 2				**α = 4**				**α = 8**				**α = 16**			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	-35.5	-98.1	-156	-199	-13.9	-36.9	-54.5	-70.3	-4.03	-10	-16.2	-21.2	-0.58	-1.47	-2.31	-3.02
	60	-35.5	-122	-174	-194	-13.9	-58.2	-72.7	-88.5	-4.03	-16.9	-22.9	-27.9	-0.58	-2.45	-3.33	-4.11
	90	-35.5	-144	-175	-178	-13.9	-69.4	-86.2	-96.4	-4.03	-20.8	-27.1	-31.8	-0.58	-3.05	-4.03	-4.86
	120	-35.5	-150	-169	-184	-13.9	-72.9	-93.6	-101	-4.03	-24.1	-30.7	-35.7	-0.58	-3.63	-4.70	-5.57
Sum		-130.3				-59.76				-18.84				-2.80			
Sum		-52.93															

Table C.26: ϕ values for sequential processes assuming different α values.

		Mixed processes – Once Payment															
		α = 2				**α = 4**				**α = 8**				**α = 16**			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	-35.5	-81.9	-147	-179	-13.9	-31.9	-55.4	-72.5	-4.03	-9.20	-16.3	-21.7	-0.58	-1.32	-2.36	-3.12
	60	-35.5	-105	-1.51	-187	-13.9	-53.5	-72	-87.6	-4.03	-16.5	-22.8	-28	-0.58	-2.43	-3.37	-4.20
	90	-35.5	-140	-167	-180	-13.9	-60	-79	-93.6	-4.03	-20.3	-27	-32.1	-0.58	-3.03	-4.06	-4.92
	120	-35.5	-147	-163	-185	-13.9	-68	-91.1	**-102**	-4.03	-23.8	-30.7	-35.9	-0.58	-3.62	-4.71	-5.60
Sum		-114.1				-57.63				-18.78				-2.82			
Sum		-48.33															

Table C.27: ϕ values for mixed processes assuming different α values.

		Random processes – Once Payment Crashing											
		λ = 0.1				**λ = 0.3**				**λ = 0.5**			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	-1	-1.19	-1.41	-1.55	-1	-1.61	-2.43	-2.99	-1	-2.24	-4.15	-5.54
	60	-1	-1.40	-1.58	-1.74	-1	-2.39	-3.10	-3.69	-1	-4.05	-5.77	-7.20
	90	-1	-1.57	-1.76	-1.92	-1	-2.88	-3.64	-4.27	-1	-5.20	-7	-8.49
	120	-1	-1.71	-1.92	-2.10	-1	-3.31	-4.12	-4.78	-1	-6.16	-8.08	-9.63
Average		-1.491				-2.700				-4.844			
Average		-3.012											

Table C.28: ϕ values for random processes assuming different λ values.

		Sequential processes – Once Payment Crashing											
		λ = 0.1				**λ = 0.3**				**λ = 0.5**			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	-1	-1.22	-1.37	-1.50	-1	-1.74	-2.33	-2.84	-1	-2.55	-4	-5.2
	60	-1	-1.40	-1.55	-1.69	-1	-2.43	-3.06	-3.60	-1	-4.24	-5.74	-7.08
	90	-1	-1.54	-1.72	-1.87	-1	-2.89	-3.59	-4.18	-1	-5.25	-6.91	-8.32
	120	-1	-1.69	-1.89	-2.06	-1	-3.33	-4.11	-4.73	-1	-6.25	-8.08	-9.51
Average		-1.469				-2.677				-4.821			
Average		-2.989											

Table C.29: ϕ values for sequential processes assuming different λ values.

		Mixed processes – Once Payment Crashing											
		λ = 0.1				**λ = 0.3**				**λ = 0.5**			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	-1	-1.19	-1.39	-1.53	-1	-1.62	-2.39	-2.93	-1	-2.27	-4.07	-5.39
	60	-1	-1.41	-1.58	-1.73	-1	-2.43	-3.10	-3.69	-1	-4.19	-5.80	-7.20
	90	-1	-1.56	-1.74	-1.90	-1	-2.90	-3.63	-4.24	-1	-5.23	-6.97	-8.42
	120	-1	-1.71	-1.92	-2.09	-1	-3.32	-4.12	-4.77	-1	-6.18	-8.10	-9.61
Average		-1.484				-2.696				-4.839			
Average		-3.006											

Table C.30: ϕ values for mixed processes assuming different λ values.

		Random processes – Single Payment Crashing											
		λ = 0.1				**λ = 0.3**				**λ = 0.5**			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	-1	-1.04	-1.07	-1.08	-1	-1.04	-1.09	-1.12	-1	-1.07	-1.15	-1.20
	60	-1	-1.08	-1.08	-1.08	-1	-1.08	-1.11	-1.13	-1	-1.14	-1.19	-1.21
	90	-1	-1.10	-1.09	-1.09	-1	-1.10	-1.12	-1.13	-1	-1.18	-1.20	-1.21
	120	-1	-1.11	-1.10	-1.10	-1	-1.11	-1.12	-1.13	-1	-1.20	-1.21	-1.21
Average		-1.064				-1.08				-1.136			
Average		-1.093											

Table C.31: ϕ values for random processes assuming different λ values.

		Sequential processes – Single Payment Crashing											
		λ = 0.1				**λ = 0.3**				**λ = 0.5**			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	-1	-1.03	-1.04	-1.04	-1	-1.02	-1.05	-1.06	-1	-1.05	-1.09	-1.13
	60	-1	-1.06	-1.05	-1.05	-1	-1.07	-1.08	-1.09	-1	-1.13	-1.15	-1.17
	90	-1	-1.07	-1.07	-1.07	-1	-1.09	-1.10	-1.11	-1	-1.16	-1.17	-1.18
	120	-1	-1.09	-1.08	-1.08	-1	-1.10	-1.11	-1.11	-1	-1.19	-1.19	-1.19
Average		-1.046				-1.062				-1.113			
Average		-1.073											

Table C.32: ϕ values for sequential processes assuming different λ values.

		Mixed processes – Single Payment Crashing											
		λ = 0.1				**λ = 0.3**				**λ = 0.5**			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	-1	-1.04	-1.06	-1.07	-1	-1.04	-1.08	-1.11	-1	-1.06	-1.14	-1.19
	60	-1	-1.08	-1.07	-1.07	-1	-1.08	-1.11	-1.13	-1	-1.15	-1.19	-1.21
	90	-1	-1.09	-1.08	-1.08	-1	-1.10	-1.11	-1.12	-1	-1.18	-1.19	-1.20
	120	-1	-1.10	-1.10	-1.10	-1	-1.11	-1.12	-1.12	-1	-1.20	-1.20	-1.20
Average		-1.059				-1.077				-1.132			
Average		-1.089											

Table C.33: ϕ values for mixed processes assuming different λ values.

		Random processes - Multiple Payment Crashing											
		λ = 0.1				**λ = 0.3**				**λ = 0.5**			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
	60	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
	90	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
	120	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
Sum		-1				-1				-1			
Sum		-1											

Table C.34: ϕ values for random processes assuming different λ values.

		Sequential processes - Multiple Payment Crashing											
		λ = 0.1				**λ = 0.3**				**λ = 0.5**			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
	60	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
	90	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
	120	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
Sum		-1				-1				-1			
Sum		-1											

Table C.35: ϕ values for sequential processes assuming different λ values.

		Mixed processes - Multiple Payment Crashing											
		λ = 0.1				**λ = 0.3**				**λ = 0.5**			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
	60	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
	90	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
	120	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
Sum		-1				-1				-1			
Sum		-1											

Table C.36: ϕ values for mixed processes assuming different λ values.

		Random processes – Once Payment Crashing											
		Front-loaded				*Medium-loaded*				*Back-loaded*			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	-1	-2.23	-4.16	-5.59	-1	-2.22	-4.07	-5.40	-1	-2.21	-3.95	-5.21
	60	-1	-3.95	-5.73	-7.33	-1	-4.12	-5.87	-7.31	-1	-3.96	-5.57	-6.90
	90	-1	-4.96	-6.85	-8.53	-1	-5.40	-7.26	-8.78	-1	-5.07	-6.80	-8.18
	120	-1	-5.84	-7.80	-9.48	-1	-6.47	-8.47	-10.0	-1	-6.03	-7.89	-9.36
Average		-4.778				-4.961				-4.695			
Average		-4.811											

Table C.37: ϕ values for the placement of bottleneck activities assuming random processes.

		Sequential processes - Once Payment Crashing											
		Front-loaded				*Medium-loaded*				*Back-loaded*			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	-1	-2.59	-4.16	-5.51	-1	-2.54	-3.99	-5.18	-1	-2.39	-3.63	-4.65
	60	-1	-4.19	-5.85	-7.38	-1	-4.37	-5.88	-7.21	-1	-3.97	-5.39	-6.62
	90	-1	-5.05	-6.84	-8.44	-1	-5.49	-7.18	-8.60	-1	-5.01	-6.68	-8
	120	-1	-5.91	-7.81	-9.41	-1	-6.53	-8.43	-9.95	-1	-6.01	-7.84	-9.19
Average		-4.821				-4.959				-4.586			
Average		-4.789											

Table C.38: ϕ values for the placement of bottleneck activities assuming sequential processes.

		Mixed processes - Once Payment Crashing											
		Front-loaded				*Medium-loaded*				*Back-loaded*			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	-1	-2.27	-4.10	-5.48	-1	-2.26	-3.96	-5.25	-1	-2.2	-3.81	-4.95
	60	-1	-4.07	-5.83	-7.44	-1	-4.24	-5.91	-7.32	-1	-3.98	-5.52	-6.84
	90	-1	-5.00	-6.85	-8.49	-1	-5.44	-7.25	-8.75	-1	-5.06	-6.74	-8.13
	120	-1	-5.82	-7.83	-9.44	-1	-6.50	-8.50	-10.0	-1	-5.99	-7.86	-9.32
Average		-4.789				-4.961				-4.65			
Average		-4.8											

Table C.39: ϕ values for the placement of bottleneck activities assuming mixed processes.

		Random processes – Single Payment Crashing											
		Front-loaded				***Medium-loaded***				***Back-loaded***			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	-1	-1.06	-1.15	-1.22	-1	-1.07	-1.15	-1.20	-1	-1.08	-1.16	-1.19
	60	-1	-1.13	-1.18	-1.22	-1	-1.15	-1.20	-1.23	-1	-1.14	-1.19	-1.20
	90	-1	-1.16	-1.19	-1.21	-1	-1.19	-1.22	-1.23	-1	-1.16	-1.19	-1.20
	120	-1	-1.18	-1.19	-1.20	-1	-1.21	-1.22	-1.23	-1	-1.18	-1.19	-1.19
Average		-1.131				-1.144				-1.129			
Average		-1.135											

Table C.40: ϕ values for the placement of bottleneck activities assuming random processes.

		Sequential processes - Single Payment Crashing											
		Front-loaded				***Medium-loaded***				***Back-loaded***			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	-1	-1.05	-1.12	-1.17	-1	-1.05	-1.10	-1.14	-1	-1.05	-1.09	-1.12
	60	-1	-1.12	-1.16	-1.18	-1	-1.13	-1.16	-1.18	-1	-1.13	-1.15	-1.16
	90	-1	-1.15	-1.16	-1.18	-1	-1.17	-1.19	-1.19	-1	-1.15	-1.16	-1.17
	120	-1	-1.17	-1.17	-1.18	-1	-1.20	-1.20	-1.20	-1	-1.16	-1.17	-1.17
Average		-1.113				-1.119				-1.105			
Average		-1.112											

Table C.41: ϕ values for the placement of bottleneck activities assuming sequential processes.

		Mixed processes - Single Payment Crashing											
		Front-loaded				***Medium-loaded***				***Back-loaded***			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	-1	-1.06	-1.15	-1.22	-1	-1.07	-1.15	-1.20	-1	-1.07	-1.14	-1.19
	60	-1	-1.13	-1.18	-1.22	-1	-1.15	-1.20	-1.22	-1	-1.14	-1.17	-1.20
	90	-1	-1.16	-1.18	-1.20	-1	-1.19	-1.21	-1.22	-1	-1.16	-1.17	-1.18
	120	-1	-1.17	-1.18	-1.19	-1	-1.21	-1.22	-1.22	-1	-1.16	-1.17	-1.18
Average		-1.128				-1.141				-1.121			
Average		-1.13											

Table C.42: ϕ values for the placement of bottleneck activities assuming mixed processes.

		Random processes – Multiple Payment Crashing											
		Front-loaded				***Medium-loaded***				***Back-loaded***			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
	60	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
	90	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
	120	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
Sum		-1				-1				-1			
Sum		-1											

Table C.43: ϕ values for the placement of bottleneck activities assuming random processes.

		Sequential processes - Multiple Payment Crashing											
		Front-loaded				***Medium-loaded***				***Back-loaded***			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
	60	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
	90	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
	120	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
Sum		-1				-1				-1			
Sum		-1											

Table C.44: ϕ values for the placement of bottleneck activities assuming sequential processes.

		Mixed processes - Multiple Payment Crashing											
		Front-loaded				***Medium-loaded***				***Back-loaded***			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
	60	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
	90	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
	120	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
Sum		-1				-1				-1			
Sum		-1											

Table C.45: ϕ values for the placement of bottleneck activities assuming mixed processes.

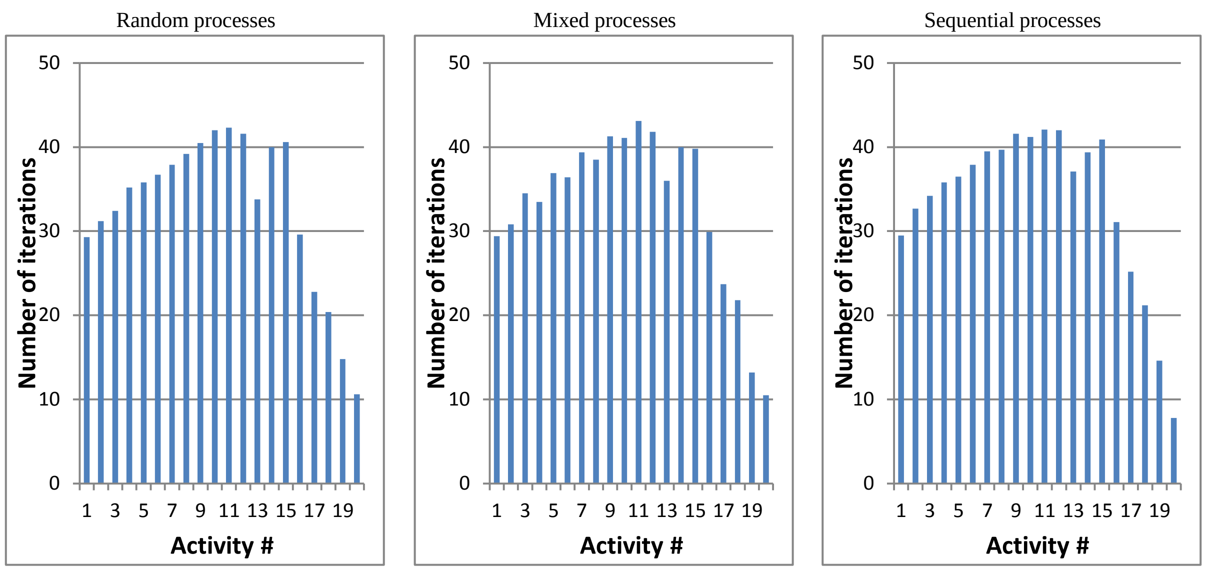


Table C.46: Distribution of reworks for back-loaded processes with n_{ff}=120 and n_{fb}=60.

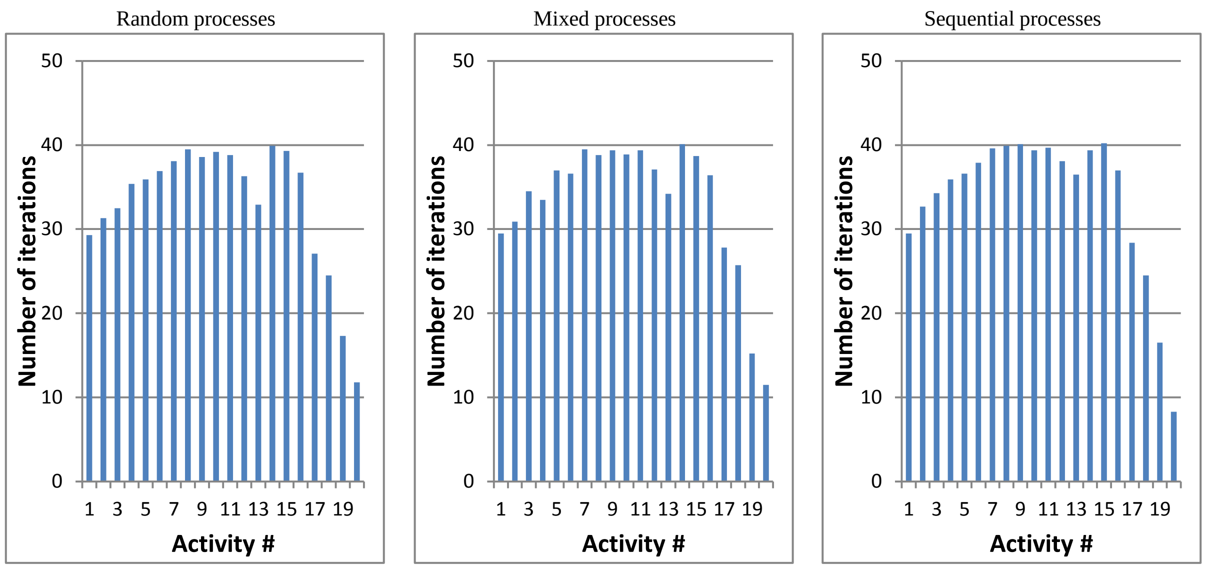


Table C.47: Distribution of reworks for medium-loaded processes with n_{ff}=120 and n_{fb}=60.

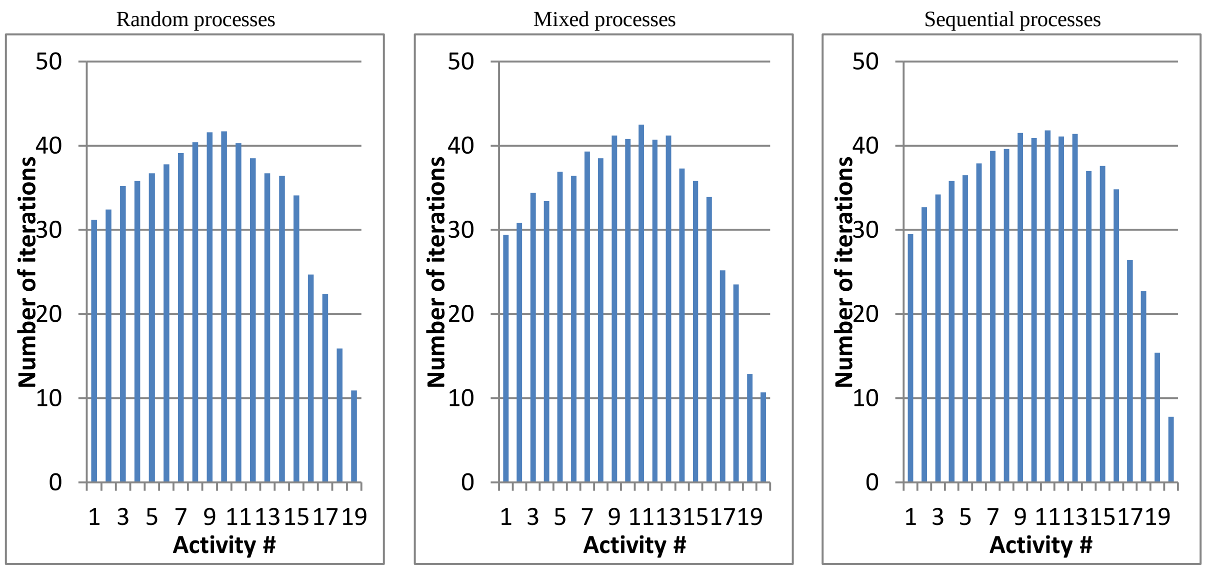


Table C.48: Distribution of reworks for equally-loaded processes with n_{ff}=120 and n_{fb}=60.

		Random processes – Once Payment Crashing											
		Feedback probability 0.1				**Feedback probability 0.3**				**Feedback probability 0.5**			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	-1	-1.05	-1.11	-1.17	-1	-1.28	-1.90	-3.12	-1	-2.15	-3.84	-4.94
	60	-1	-1.07	-1.14	-1.24	-1	-1.75	-4.18	-6.02	-1	-3.84	-5.46	-6.86
	90	-1	-1.08	-1.18	-1.32	-1	-2.92	-5.96	-7.53	-1	-4.86	-6.68	-8.37
	120	-1	-1.09	-1.23	-1.42	-1	-4.17	-6.83	-8.57	-1	-5.64	-7.76	-9.63
Average		-1.131				-3.639				-4.627			
Average		-3.133											

Table C.49: ϕ values for the base case of P_4 assuming random processes.

		Sequential processes – Once Payment Crashing											
		Feedback probability 0.1				**Feedback probability 0.3**				**Feedback probability 0.5**			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	-1	-1.06	-1.13	-1.21	-1	-1.34	-1.97	-2.91	-1	-2.44	-3.84	-4.96
	60	-1	-1.07	-1.16	-1.26	-1	-1.93	-4.46	-6.08	-1	-4.07	-5.55	-6.93
	90	-1	-1.08	-1.19	-1.33	-1	-3.21	-6.05	-7.50	-1	-4.95	-6.70	-8.30
	120	-1	-1.10	-1.23	-1.43	-1	-4.32	-6.85	-8.56	-1	-5.69	-7.71	-9.57
Average		-1.141				-3.699				-4.669			
Average		-3.170											

Table C.50: ϕ values for the base case of P_4 assuming sequential processes.

		Mixed processes – Once Payment Crashing											
		Feedback probability 0.1				**Feedback probability 0.3**				**Feedback probability 0.5**			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	-1	-1.05	-1.11	-1.17	-1	-1.28	-1.83	-2.87	-1	-2.18	-3.75	-4.80
	60	-1	-1.07	-1.14	-1.24	-1	-1.80	-4.27	-6.08	-1	-3.95	-5.49	-6.87
	90	-1	-1.08	-1.18	-1.32	-1	-3.07	-6.01	-7.52	-1	-4.90	-6.70	-8.37
	120	-1	-1.09	-1.23	-1.43	-1	-4.21	-6.87	-8.58	-1	-5.68	-7.79	-9.64
Average		-1.132				-3.649				-4.633			
Average		-3.138											

Table C.51: ϕ values for the base case of P_4 assuming mixed processes.

		Random processes – Single Payment Crashing											
		Feedback probability 0.1				**Feedback probability 0.3**				**Feedback probability 0.5**			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	-1	-1	-1.01	-1.01	-1	-1.01	-1.03	-1.04	-1	-1.04	-1.08	-1.13
	60	-1	-1	-1	-1	-1	-1.02	-1.05	-1.08	-1	-1.10	-1.14	-1.17
	90	-1	-1	-1	-1.01	-1	-1.04	-1.09	-1.12	-1	-1.13	-1.17	-1.19
	120	-1	-1	-1	-1.01	-1	-1.07	-1.11	-1.14	-1	-1.15	-1.18	-1.20
Average		-1.003				-1.05				-1.105			
Average		-1.053											

Table C.52: ϕ values for the base case of P_4 assuming random processes.

		Sequential processes – Single Payment Crashing											
		Feedback probability 0.1				**Feedback probability 0.3**				**Feedback probability 0.5**			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	-1	-1	-1	-1	-1	-1	-1.01	-1.02	-1	-1.02	-1.05	-1.08
	60	-1	-1	-1	-1	-1	-1.01	-1.04	-1.07	-1	-1.09	-1.11	-1.14
	90	-1	-1	-1	-1	-1	-1.04	-1.09	-1.10	-1	-1.12	-1.14	-1.16
	120	-1	-1	-1	-1	-1	-1.07	-1.11	-1.13	-1	-1.15	-1.16	-1.18
Average		-1				-1.043				-1.088			
Average		-1.044											

Table C.53: ϕ values for the base case of P_4 assuming sequential processes.

		Mixed processes – Single Payment Crashing											
		Feedback probability 0.1				**Feedback probability 0.3**				**Feedback probability 0.5**			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	-1	-1	-1.01	-1.01	-1	-1.01	-1.02	-1.04	-1	-1.04	-1.08	-1.12
	60	-1	-1	-1	-1	-1	-1.02	-1.05	-1.08	-1	-1.09	-1.14	-1.16
	90	-1	-1	-1	-1.01	-1	-1.04	-1.09	-1.11	-1	-1.13	-1.16	-1.19
	120	-1	-1	-1	-1.01	-1	-1.07	-1.11	-1.14	-1	-1.16	-1.18	-1.20
Average		-1.003				-1.049				-1.103			
Average		-1.051											

Table C.54: ϕ values for the base case of P_4 assuming mixed processes.

		Random processes – Once Payment Crashing											
		Feedback probability 0.1				**Feedback probability 0.3**				**Feedback probability 0.5**			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	-1	-1.05	-1.11	-1.18	-1	-1.29	-1.95	-3.43	-1	-2.28	-4.24	-5.68
	60	-1	-1.07	-1.15	-1.25	-1	-1.86	-4.66	-6.80	-1	-4.34	-6.33	-8
	90	-1	-1.08	-1.19	-1.33	-1	-3.32	-6.95	-8.87	-1	-5.86	-7.96	-9.84
	120	-1	-1.10	-1.24	-1.44	-1	-5.20	-8.69	-10.6	-1	-7.13	-9.40	-11.4
Average		-1.137				-4.226				-5.404			
Average		-3.589											

Table C.55: ϕ values for the base case of P_5 assuming random processes.

		Sequential processes – Once Payment Crashing											
		Feedback probability 0.1				**Feedback probability 0.3**				**Feedback probability 0.5**			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	-1	-1.06	-1.13	-1.22	-1	-1.36	-2.05	-3.14	-1	-2.54	-4.02	-5.27
	60	-1	-1.07	-1.16	-1.27	-1	-2.07	-4.85	-6.66	-1	-4.46	-6.08	-7.58
	90	-1	-1.08	-1.19	-1.34	-1	-3.64	-6.98	-8.67	-1	-5.77	-7.65	-9.33
	120	-1	-1.10	-1.24	-1.46	-1	-5.44	-8.56	-10.4	-1	-7.08	-9.18	-10.9
Average		-1.145				-4.239				-5.241			
Average		-3.541											

Table C.56: ϕ values for the base case of P_5 assuming sequential processes.

		Mixed processes – Once Payment Crashing											
		Feedback probability 0.1				**Feedback probability 0.3**				**Feedback probability 0.5**			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	-1	-1.05	-1.11	-1.18	-1	-1.29	-1.90	-3.14	-1	-2.32	-4.15	-5.49
	60	-1	-1.07	-1.15	-1.25	-1	-1.93	-4.67	-6.75	-1	-4.46	-6.33	-7.96
	90	-1	-1.08	-1.19	-1.33	-1	-3.50	-6.94	-8.82	-1	-5.85	-7.87	-9.69
	120	-1	-1.10	-1.24	-1.45	-1	-5.32	-8.62	-10.6	-1	-7.12	-9.32	-11.3
Average		-1.138				-4.218				-5.366			
Average		-3.574											

Table C.57: ϕ values for the base case of P_5 assuming mixed processes.

		Random processes – Single Payment Crashing											
		Feedback probability 0.1				**Feedback probability 0.3**				**Feedback probability 0.5**			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	-1	-1	-1.01	-1.01	-1	-1.02	-1.04	-1.08	-1	-1.07	-1.16	-1.22
	60	-1	-1	-1	-1.01	-1	-1.04	-1.11	-1.16	-1	-1.17	-1.25	-1.30
	90	-1	-1	-1.01	-1.01	-1	-1.09	-1.18	-1.23	-1	-1.25	-1.31	-1.35
	120	-1	-1	-1.01	-1.02	-1	-1.16	-1.24	-1.28	-1	-1.30	-1.35	-1.39
Average		-1.001				-1.102				-1.195			
Average		-1.100											

Table C.58: ϕ values for the base case of P_5 assuming random processes.

		Sequential processes – Single Payment Crashing											
		Feedback probability 0.1				**Feedback probability 0.3**				**Feedback probability 0.5**			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	-1	-1	-1	-1	-1	-1	-1.02	-1.05	-1	-1.05	-1.10	-1.14
	60	-1	-1	-1	-1	-1	-1.03	-1.10	-1.14	-1	-1.15	-1.19	-1.22
	90	-1	-1	-1	-1	-1	-1.10	-1.17	-1.21	-1	-1.22	-1.25	-1.28
	120	-1	-1	-1	-1	-1	-1.16	-1.23	-1.26	-1	-1.28	-1.31	-1.34
Average		-1				-1.092				-1.158			
Average		-1.083											

Table C.59: ϕ values for the base case of P_5 assuming sequential processes.

		Mixed processes – Single Payment Crashing											
		Feedback probability 0.1				**Feedback probability 0.3**				**Feedback probability 0.5**			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	-1	-1	-1.01	-1.01	-1	-1.01	-1.04	-1.07	-1	-1.07	-1.14	-1.21
	60	-1	-1	-1	-1.01	-1	-1.03	-1.11	-1.16	-1	-1.17	-1.24	-1.29
	90	-1	-1	-1	-1.01	-1	-1.09	-1.18	-1.22	-1	-1.24	-1.29	-1.33
	120	-1	-1	-1.01	-1.02	-1	-1.16	-1.24	-1.27	-1	-1.30	-1.34	-1.38
Average		-1				-1.10				-1.19			
Average		-1.10											

Table C.60: ϕ values for the base case of P_5 assuming mixed processes.

		Random processes – Multi Payment Crashing											
		Feedback probability 0.1				**Feedback probability 0.3**				**Feedback probability 0.5**			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	0	1.4	3.6	6.1	0	10.5	42.5	132	0	70.7	211	305
	60	0	2	5.3	9.1	0	38	219	377	0	198	341	461
	90	0	2.7	6.8	12.4	0	114	354	491	0	276	438	568
	120	0	3.2	8.7	16.8	0	202	437	591	0	341	520	663
Average		4.881				188				274.5			
Average		155.8											

Table C.61: Number reworks for the base case of P_{3a} assuming random processes.

		Sequential processes – Multi Payment Crashing											
		Feedback probability 0.1				**Feedback probability 0.3**				**Feedback probability 0.5**			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	0	2	5	8.1	0	14.8	50	117	0	97.3	214	308
	60	0	2.2	6	10.3	0	50.8	240	376	0	224	360	478
	90	0	3	7.2	13.1	0	137	361	447	0	294	451	581
	120	0	3.3	8.9	17.4	0	220	446	597	0	356	533	676
Average		5.41				191				285.8			
Average		160.7											

Table C.62: Number reworks for the base case of P_{3a} assuming sequential processes.

		Mixed processes – Multi Payment Crashing											
		Feedback probability 0.1				**Feedback probability 0.3**				**Feedback probability 0.5**			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	0	1.4	3.6	6.1	0	11	40	111	0	73	203	290
	60	0	2	5.3	9.3	0	40.9	224	376	0	209	346	463
	90	0	2.8	6.9	12.6	0	124	357	492	0	283	442	572
	120	0	3.3	8.7	17	0	208	441	594	0	346	526	668
Average		4.94				188.7				276.3			
Average		156.6											

Table C.63: Number reworks values for the base case of P_{3a} assuming mixed processes.

D: Simulation results for section 5.3.3

		Random processes											
		Feedback probability 0.1				**Feedback probability 0.3**				**Feedback probability 0.5**			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	-1.20	-1.23	-1.23	-1.24	-1.20	-1.27	-1.44	-1.76	-1.20	-1.39	-1.04	-1.16
	60	-1.05	-1.07	-1.09	-1.10	-1.05	-1.33	-1.40	-1.04	-1.05	-1.05	-1.20	-1.33
	90	-1.01	-1.04	-1.08	-1.14	-1.01	-1.51	-1.04	-1.08	-1.01	-1.15	-1.37	-1.42
	120	-1.00	-1.06	-1.12	-1.23	-1.00	-1.15	-1.07	-1.23	-1.00	-1.27	-1.53	-1.60
Average		-1.118				-1.224				-1.236			
Average		-1.191											

Table D.64: ψ values for the base case of P_{3b} assuming random processes.

		Sequential processes											
		Feedback probability 0.1				**Feedback probability 0.3**				**Feedback probability 0.5**			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	-1	-0.96	-0.93	-0.91	-1	-0.91	-0.99	-1.17	-1	-1.04	-0.96	-0.98
	60	-1	-0.98	-0.96	-0.96	-1	-1.20	-1.25	-1.02	-1	-1.05	-1.18	-1.22
	90	-1	-0.99	-1.00	-1.04	-1	-1.35	-1.00	-1.08	-1	-1.18	-1.35	-1.41
	120	-1	-1.02	-1.07	-1.15	-1	-1.08	-1.09	-1.25	-1	-1.27	-1.51	-1.58
Average		-0.998				-1.087				-1.171			
Average		-1.085											

Table D.65: ψ values for the base case of P_{3b} assuming sequential processes.

		Mixed processes											
		Feedback probability 0.1				**Feedback probability 0.3**				**Feedback probability 0.5**			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	-1.14	-1.14	-1.13	-1.14	-1.14	-1.15	-1.32	-1.66	-1.14	-1.30	-1.03	-1.11
	60	-1.03	-1.03	-1.04	-1.05	-1.03	-1.30	-1.35	-1.04	-1.03	-1.04	-1.20	-1.30
	90	-1.02	-1.03	-1.06	-1.12	-1.02	-1.47	-1.00	-1.07	-1.02	-1.14	-1.36	-1.42
	120	-1.01	-1.05	-1.11	-1.21	-1.01	-1.14	-1.08	-1.24	-1.01	-1.28	-1.53	-1.59
Average		-1.081				-1.189				-1.219			
Average		-1.163											

Table D.66: ψ values for the base case of P_{3b} assuming mixed processes.

		Random processes											
		Feedback probability 0.1				Feedback probability 0.3				Feedback probability 0.5			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	0	1.40	3.63	6.09	0	10.7	42.9	134	0	70.8	211	306
	60	0	2.05	5.24	9.10	0	38.3	220	378	0	201	343	463
	90	0	2.75	6.84	12.4	0	116	356	491	0	278	439	569
	120	0	3.27	8.70	16.9	0	199	439	591	0	340	523	664
Average		4.898				188				275			
Average		156											

Table D.67: Number reworks before overlapping for the base case of P_{3b}.

		Sequential processes											
		Feedback probability 0.1				Feedback probability 0.3				Feedback probability 0.5			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	0	2.00	4.97	8.23	0	14.7	49.5	117	0	96.5	213	307
	60	0	2.25	5.97	10.3	0	50.9	242	377	0	225	362	479
	90	0	2.91	7.17	13.1	0	137	360	494	0	293	450	579
	120	0	3.30	8.91	17.2	0	215	443	594	0	353	530	673
Average		5.394				193				285			
Average		161.1											

Table D.68: Number reworks before overlapping for the base case of P_{3b}.

		Mixed processes											
		Feedback probability 0.1				Feedback probability 0.3				Feedback probability 0.5			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	0	1.36	3.55	6.01	0	10.5	39.0	111	0	69.9	201	289
	60	0	2.08	5.37	9.33	0	41.6	224	375	0	209	346	464
	90	0	2.79	6.90	12.5	0	123	357	493	0	283	444	573
	120	0	3.30	8.74	17	0	208	439	593	0	346	524	666
Average		4.933				188				276			
Average		156.3											

Table D.69: Number reworks before overlapping for the base case of P_{3b}.

		Random processes											
		Feedback probability 0.1				Feedback probability 0.3				Feedback probability 0.5			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	0	1.38	3.62	6.08	0	10.4	39.4	118	0	64.7	206	296
	60	0	2.02	5.22	8.99	0	34.7	203	374	0	198	337	446
	90	0	2.72	6.74	12.1	0	105	360	504	0	275	424	546
	120	0	3.21	8.52	16.1	0	196	451	596	0	334	498	626
Average		4.793				187				266			
Average		152.6											

Table D.70: Number reworks after overlapping for the base case of P_{3b}.

		Sequential processes											
		Feedback probability 0.1				Feedback probability 0.3				Feedback probability 0.5			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	0	2	4.97	8.23	0	14.6	47.1	106	0	90.8	205	296
	60	0	2.26	5.94	10.2	0	46.4	227	375	0	224	357	470
	90	0	2.91	7.11	12.8	0	126	370	506	0	291	440	562
	120	0	3.29	8.68	16.6	0	215	455	599	0	349	509	638
Average		5.31				193				276			
Average		158.1											

Table D.71: Number reworks after overlapping for the base case of P_{3b}.

		Mixed processes											
		Feedback probability 0.1				Feedback probability 0.3				Feedback probability 0.5			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	0	1.36	3.52	5.96	0	1.39	36	97.5	0	63.9	194	278
	60	0	2.08	5.31	9.20	0	37.6	207	372	0	207	340	449
	90	0	2.77	6.80	12.2	0	112	364	507	0	281	430	552
	120	0	3.27	8.54	16.2	0	205	450	596	0	339	499	629
Average		4.825				186.6				266.4			
Average		152.6											

Table D.72: Number reworks after overlapping for the base case of P_{3b}.

		Random processes											
		Feedback probability 0.1				**Feedback probability 0.3**				**Feedback probability 0.5**			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	0	0	0.06	0.68	0	1.43	9.23	45.7	0	20.8	119	246
	60	0	0	0	0.3	0	7.8	90	211	0	126	248	368
	90	0	0	0	0.4	0	48.4	210	300	0	232	348	452
	120	0	0	0	0.9	0	125	295	378	0	338	450	550
Average		0.146				108				219			
Average		109											

Table D.73: Number partial reworks before applying overlapping for the base case of P_{3b}.

		Sequential processes											
		Feedback probability 0.1				**Feedback probability 0.3**				**Feedback probability 0.5**			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	0	0	0	0	0	0.01	5.6	28	0	19.6	76.7	150
	60	0	0	0	0	0	9.1	89.3	177	0	115	188	265
	90	0	0	0	0	0	56.6	187	255	0	207	287	362
	120	0	0	0	0.3	0	134	273	342	0	322	400	476
Average		0.019				97.3				179			
Average		92.2											

Table D.74: Number partial reworks before applying overlapping for the base case of P_{3b}.

		Mixed processes											
		Feedback probability 0.1				**Feedback probability 0.3**				**Feedback probability 0.5**			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	0	0	0	0.5	0	1.2	7.9	36.6	0	19.4	112	238
	60	0	0	0	0.1	0	8.2	89.8	204	0	129	238	351
	90	0	0	0	0.1	0	51.7	205	292	0	227	336	435
	120	0	0	0	0.8	0	133	290	370	0	342	443	532
Average		0.094				106				213			
Average		106											

Table D.75: Number partial reworks before applying overlapping for the base case of P_{3b}.

		Random processes											
		Feedback probability 0.1				**Feedback probability 0.3**				**Feedback probability 0.5**			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	0	0	0.07	0.72	0	1.72	9.46	42.4	0	21.4	128	284
	60	0	0	0.01	0.54	0	8.48	89.0	238	0	155	367	614
	90	0	0	0	0.93	0	50	267	468	0	346	627	928
	120	0	0	0.67	1.39	0	152	468	698	0	550	878	1211
Average		0.271				156				382			
Average		179.4											

Table D.76: Number partial reworks after applying overlapping for the base case of P_{3b}.

		Sequential processes											
		Feedback probability 0.1				**Feedback probability 0.3**				**Feedback probability 0.5**			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	0	0	0	0	0	0.2	5.69	26.4	0	20.2	86.5	186
	60	0	0	0	0	0	9.36	90.8	212	0	149	282	444
	90	0	0	0	0	0	59.7	259	420	0	313	500	711
	120	0	0	0	1	0	167	442	631	0	509	756	1022
Average		0.063				145				311			
Average		152											

Table D.77: Number partial reworks after applying overlapping for the base case of P_{3b}.

		Mixed processes											
		Feedback probability 0.1				**Feedback probability 0.3**				**Feedback probability 0.5**			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	0	0	0.02	0.41	0	1.39	8.09	33.9	0	19.6	118	263
	60	0	0	0	0.29	0	8.80	89.2	232	0	159	356	589
	90	0	0	0	0.72	0	53.5	267	460	0	342	601	877
	120	0	0	0.41	1.12	0	159	461	672	0	544	848	1161
Average		0.189				153				367			
Average		173.4											

Table D.78: Number partial reworks after applying overlapping for the base case of P_{3b}.

		Random processes											
		Feedback probability 0.1				Feedback probability 0.3				Feedback probability 0.5			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	14.8	15.8	16.7	17.4	14.8	17.8	17.3	14.0	14.8	16.9	19.3	18.9
	60	18.5	19.5	20.4	21.2	18.5	18.7	17.1	22.0	18.5	22.6	25.0	25.4
	90	19.8	20.9	21.8	22.2	19.8	17.3	26.1	30.1	19.8	26.3	27.1	28.5
	120	20.4	21.5	22.3	22.2	20.4	23.6	31.2	31.5	20.4	27.3	27.5	28.5
Average		19.71				21.26				22.95			
Average		21.31											

Table D.79: Percental change in process cost due to overlapping for the base case of P_{3b}.

		Sequential processes											
		Feedback probability 0.1				Feedback probability 0.3				Feedback probability 0.5			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	20.9	22.2	23.2	24	20.9	24.1	23.2	19.9	20.9	21.9	23.4	23.7
	60	20.9	22.2	23.2	24	20.9	20.2	19.4	24.6	20.9	25.3	26.3	27.4
	90	20.9	22.2	23.2	23.7	20.9	19.2	28.9	31.0	20.9	26.7	27.5	28.8
	120	20.9	22.3	23.0	23.1	20.9	25.5	31.5	31.3	20.9	27.3	27.7	28.7
Average		22.49				23.9				24.89			
Average		23.76											

Table D.80: Percental change in process cost due to overlapping for the base case of P_{3b}.

		Mixed processes											
		Feedback probability 0.1				Feedback probability 0.3				Feedback probability 0.5			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	16.3	17.2	18.1	18.8	16.3	19.3	18.4	14.7	16.3	17.7	19.0	18.2
	60	19.3	20.4	21.3	22.0	19.3	19.0	17.7	22.5	19.3	23.3	25.5	25.9
	90	20.0	21.2	22.1	22.4	20.0	17.7	27.1	30.5	20.0	26.8	27.4	28.7
	120	20.3	21.6	22.4	22.3	20.3	23.7	31.1	31.2	20.3	26.9	27.2	28.4
Average		20.36				21.8				23.18			
Average		21.78											

Table D.81: Percental change in process cost due to overlapping for the base case of P_{3b}.

		Random processes											
		Feedback probability 0.1				Feedback probability 0.3				Feedback probability 0.5			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	-17.8	-19.4	-20.6	-21.7	-17.8	-22.6	-24.9	-24.7	-17.8	-23.5	-20.0	-22.0
	60	-19.4	-20.8	-22.1	-23.3	-19.4	-24.8	-23.8	-22.9	-19.4	-23.6	-30.1	-33.8
	90	-20.0	-21.8	-23.6	-25.3	-20.0	-26.2	-27.1	-32.6	-20.0	-30.3	-37.3	-40.9
	120	-20.4	-22.7	-25.1	-27.3	-20.4	-27.0	-33.5	-38.6	-20.4	-34.1	-41.6	-45.1
Average		-21.96				-25.39				-28.74			
Average		-25.36											

Table D.82: Percental change in process duration due to overlapping for the base case of P_{3b}.

		Sequential processes											
		Feedback probability 0.1				Feedback probability 0.3				Feedback probability 0.5			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	-20.9	-21.2	-21.5	-21.7	-20.9	-22.0	-22.9	-23.3	-20.9	-22.7	-22.5	-23.1
	60	-20.9	-21.6	-22.3	-23.0	-20.9	-24.2	-24.4	-25.1	-20.9	-26.4	-30.9	-33.5
	90	-20.9	-22.1	-23.4	-24.7	-20.9	-25.9	-28.9	-33.7	-20.9	-31.6	-37.2	-40.5
	120	-20.9	-22.7	-24.7	-26.7	-20.9	-27.5	-34.5	-39.2	-20.9	-34.7	-41.7	-45.5
Average		-22.45				-25.95				-29.61			
Average		-26											

Table D.83: Percental change in process duration due to overlapping for the base case of P_{3b}.

		Mixed processes											
		Feedback probability 0.1				Feedback probability 0.3				Feedback probability 0.5			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	-18.5	-19.6	-20.6	-21.4	-18.5	-22.1	-24.3	-24.5	-18.5	-23.1	-19.6	-20.2
	60	-19.9	-21.0	-22.1	-23.2	-19.9	-24.7	-23.9	-23.4	-19.9	-24.3	-30.5	-33.8
	90	-20.3	-21.9	-23.4	-25.1	-20.3	-26.0	-27.4	-32.8	-20.3	-30.6	-37.1	-40.8
	120	-20.5	-22.6	-24.8	-27.1	-20.5	-27.0	-33.5	-38.8	-20.5	-34.3	-41.6	-45.2
Average		-22				-25.48				-28,8			
Average		-25.43											

Table D.84: Percental change in process duration due to overlapping for the base case of P_{3b}.

		Random processes											
		Alpha=1				Alpha=0.25				Alpha=0			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	0	-0.08	-0.08	-0.17	-3.55	-4.54	-2.85	-2.92	-inf.	14.8	-71.7	-27.3
	60	0	-0.14	-0.33	-0.44	-3.16	-2.50	-2.33	-2.40	-inf.	-12	-6.16	-6
	90	0	-0.30	-0.49	-0.59	-3.03	-2.19	-2.36	-2.31	-inf.	-5.30	-4.87	-4.29
	120	0	-0.40	-0.60	-0.70	-3.01	-2.21	-2.48	-2.47	-inf.	-4.58	-4.70	-4.35
Average		-0.27				-2.769				-infinity			
Average		-infinity											

Table D.85: ψ values for different alpha values assuming a feedback probability 0.5.

		Sequential processes											
		Alpha=1				Alpha=0.25				Alpha=0			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	0	-0.04	-0.06	-0.09	-3	-3.28	-2.68	-2.52	-inf.	26.7	-51.1	-19.2
	60	0	-0.17	-0.29	-0.36	-3	-2.37	-2.37	-2.30	-inf.	-8.69	-6.73	-5.60
	90	0	-0.30	-0.48	-0.57	-3	-2.29	-2.42	-2.37	-inf.	-5.67	-5.36	-4.73
	120	0	-0.40	-0.59	-0.68	-3	-2.29	-2.54	-2.54	-inf.	-4.91	-5.04	-4.68
Average		-0.252				-2.623				-infinity			
Average		-infinity											

Table D.86: ψ values for different alpha values assuming a feedback probability 0.5.

		Mixed processes											
		Alpha=1				Alpha=0.25				Alpha=0			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	0	-0.06	-0.05	-0.11	-3.38	-4.31	-3.00	-2.98	-inf.	14.9	87.5	-141
	60	0	-0.15	-0.32	-0.43	-3.08	-2.43	-2.32	-2.38	-inf.	-10.8	-6.15	-5.60
	90	0	-0.30	-0.48	-0.57	-3.05	-2.17	-2.35	-2.32	-inf.	-5.12	-4.88	-4.34
	120	0	-0.40	-0.59	-0.68	-3.02	-2.28	-2.53	-2.50	-inf.	-4.79	-4.91	-4.46
Average		-0.259				-2.756				-infinity			
Average		-infinity											

Table D.87: ψ values for different alpha values assuming a feedback probability 0.5.

		Random processes															
		Lambda=0.9				**Lambda=0.7**				**Lambda=0.3**				**Lambda=0.1**			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	-1.20	-1.63	-1.08	-1.07	-1.20	-1.45	-1.03	-1.10	-1.20	-1.23	-0.98	-1.08	-1.20	-0.99	-0.79	-0.82
	60	-1.05	-1.03	-0.96	-0.97	-1.05	-1.00	-1.06	-1.12	-1.05	-1.08	-1.24	-1.40	-1.05	-0.84	-0.86	-0.91
	90	-1.01	-1.08	-1.04	-0.98	-1.01	-1.04	-1.14	-1.13	-1.01	-1.24	-1.49	-1.65	-1.01	-0.91	-0.97	-1.04
	120	-1.00	-1.16	-1.13	-1.08	-1.00	-1.11	-1.23	-1.23	-1.00	-1.40	-1.71	-1.89	-1.00	-1.02	-1.12	-1.19
Sum		-1.092				-1.119				-1.291				-0.983			
Sum		-1.121															

Table D.88: ψ values for different lambda values assuming a feedback probability 0.5.

		Sequential processes															
		Lambda=0.9				**Lambda=0.7**				**Lambda=0.3**				**Lambda=0.1**			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	-1	-1.33	-1.10	-1.10	-1	-1.15	-1.02	-1.04	-1	-0.89	-0.82	-0.81	-1	-0.75	-0.64	-0.58
	60	-1	-0.99	-1.03	-1.02	-1	-1	-1.10	-1.13	-1	-1	-1.05	-1.08	-1	-0.78	-0.71	-0.68
	90	-1	-1.12	-1.12	-1.06	-1	-1.09	-1.19	-1.19	-1	-1.20	-1.31	-1.39	-1	-0.86	-0.83	-0.83
	120	-1	-1.17	-1.19	-1.14	-1	-1.14	-1.27	-1.28	-1	-1.36	-1.57	-1.71	-1	-0.97	-0.99	-1.02
Sum		-1.086				-1.1				-1.137				-0.853			
Sum		-1.044															

Table D.89: ψ values for different lambda values assuming a feedback probability 0.5.

		Mixed processes															
		Lambda=0.9				**Lambda=0.7**				**Lambda=0.3**				**Lambda=0.1**			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	-1.14	-1.57	-1.12	-1.05	-1.14	-1.38	-1.03	-1.07	-1.14	-1.14	-0.95	-1.00	-1.14	-0.92	-0.75	-0.76
	60	-1.03	-1.02	-0.99	-0.99	-1.03	-1	-1.07	-1.13	-1.03	-1.04	-1.19	-1.31	-1.03	-0.82	-0.82	-0.84
	90	-1.02	-1.08	-1.05	-1	-1.02	-1.05	-1.15	-1.14	-1.02	-1.22	-1.44	-1.58	-1.02	-0.91	-0.93	-0.97
	120	-1.01	-1.17	-1.17	-1.12	-1.01	-1.13	-1.25	-1.25	-1.01	-1.41	-1.69	-1.85	-1.01	-1.01	-1.09	-1.14
Sum		-1.096				-1.116				-1.251				-0.948			
Sum		-1.103															

Table D.90: ψ values for different lambda values assuming a feedback probability 0.5.

		Random processes											
		Front-loaded				***Medium-loaded***				***Back-loaded***			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	-1.14	-1.30	-1.07	-1.36	-1.22	-1.38	-1.04	-1.14	-1.21	-1.49	-1.03	-1.02
	60	-1.02	-0.97	-1.17	-1.42	-1.06	-1.11	-1.30	-1.42	-1.06	-1.17	-1.20	-1.25
	90	-0.99	-1.05	-1.25	-1.36	-1.00	-1.27	-1.53	-1.63	-1.02	-1.24	-1.37	-1.43
	120	-1.00	-1.13	-1.35	-1.43	-1.00	-1.39	-1.68	-1.78	-1.01	-1.30	-1.52	-1.63
Sum		-1.188				-1.309				-1.247			
Sum		-1.248											

Table D.91: ψ values for the placement of bottleneck activities assuming random processes.

		Sequential processes											
		Front-loaded				***Medium-loaded***				***Back-loaded***			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	-1.00	-1.04	-1.05	-1.18	-1.00	-1.07	-0.96	-0.95	-1.00	-1.03	-0.97	-0.96
	60	-1.00	-0.99	-1.16	-1.31	-1.00	-1.08	-1.19	-1.26	-1.00	-1.16	-1.20	-1.18
	90	-1.00	-1.08	-1.27	-1.34	-1.00	-1.23	-1.42	-1.50	-1.00	-1.31	-1.35	-1.39
	120	-1.00	-1.15	-1.37	-1.44	-1.00	-1.29	-1.61	-1.75	-1.00	-1.35	-1.51	-1.62
Sum		-1.149				-1.207				-1.189			
Sum		-1.182											

Table D.92: ψ values for the placement of bottleneck activities assuming sequential processes.

		Mixed processes											
		Front-loaded				***Medium-loaded***				***Back-loaded***			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	-1.09	-1.23	-1.10	-1.30	-1.12	-1.33	-1.05	-1.07	-1.14	-1.33	-0.98	-0.95
	60	-1.02	-0.99	-1.19	-1.43	-1.02	-1.11	-1.28	-1.40	-1.04	-1.15	-1.20	-1.24
	90	-1.01	-1.04	-1.26	-1.36	-1.02	-1.26	-1.51	-1.61	-1.02	-1.25	-1.35	-1.42
	120	-1.01	-1.16	-1.37	-1.44	-1.01	-1.35	-1.66	-1.77	-1.01	-1.34	-1.54	-1.64
Sum		-1.188				-1.286				-1.225			
Sum		-1.233											

Table D.93: ψ values for the placement of bottleneck activities assuming mixed processes.

		Random processes															
		Lambda=0.9				Lambda=0.7				Lambda=0.3				Lambda=0.1			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	15	14.3	17	17.1	15	16.1	18.5	18.2	15	18	20.1	20.2	15	19.5	22.9	22.6
	60	18.4	20.5	24.7	25.8	18.4	22.5	26.2	27.2	18.4	21.6	22.8	22.7	18.4	23.9	24.3	23.9
	90	19.8	23.4	26.5	28.8	19.8	27	29.3	31.7	19.8	23.1	23	23.1	19.8	24.7	24.6	24.3
	120	20.3	23	26.4	28.5	20.3	27.4	29.7	31.6	20.3	23.6	23.1	23	20.3	24.4	24.4	24.4
Sum		21.84				23.68				21.11				22.33			
Sum		22.24															

Table D.94: Percental change in process cost for different lambda assuming feedback probability 0.5.

		Sequential processes															
		Lambda=0.9				Lambda=0.7				Lambda=0.3				Lambda=0.1			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	20.9	16.8	17.4	16	20.9	19.7	20.6	19.7	20.9	25	27.3	28.2	20.9	28.2	32.7	35.5
	60	20.9	24.1	26.3	27.3	20.9	26.1	28	29.3	20.9	23.9	24.8	25.3	20.9	27	29.2	30.8
	90	20.9	24.4	26.9	29.3	20.9	28	29.9	32.1	20.9	23.4	24.2	24.9	20.9	25.8	27.4	28.6
	120	20.9	23.8	26.6	28.8	20.9	27.9	30	31.9	20.9	23.5	23.8	24.2	20.9	25.3	26.3	27.1
Sum		23.21				25.43				23.88				26,72			
Sum		24.81															

Table D.95: Percental change in process cost for different lambda assuming feedback probability 0.5.

		Mixed processes															
		Lambda=0.9				Lambda=0.7				Lambda=0.3				Lambda=0.1			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	16.4	14.7	16.1	16	16.4	16.6	18	17.2	16.4	19.3	20.8	20.7	16.4	21.3	24.7	24.6
	60	19.3	21.3	25	25.9	19.3	23.3	26.7	27.5	19.3	22.4	23.3	23.8	19.3	24.9	25.4	25.5
	90	20	23.9	26.6	28.8	20	27.5	29.6	31.8	20	23.3	23.3	23.6	20	24.7	25.2	25.4
	120	20.3	23.4	26.5	28.6	20.3	27.2	29.4	31.5	20.3	23.4	23.1	23.2	20.3	24.6	24.8	25.1
Sum		22.05				23.89				21.64				23.26			
Sum		22.71															

Table D.96: Percental change in process cost for different lambda assuming feedback probability 0.5.

		Random processes															
		Lambda=0.9				Lambda=0.7				Lambda=0.3				Lambda=0.1			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*				*Number Feedbacks*			
		0	20	40	60	0	20	40	60	0	20	40	60	0	20	40	60
Number Feed-forwards	30	-17.9	-23.4	-18.3	-18.3	-17.9	-23.3	-19.1	-20	-17.9	-22.1	-19.7	-21.8	-17.9	-19.4	-18.1	-18.5
	60	-19.4	-21.1	-23.8	-24.9	-19.4	-22.5	-27.7	-30.3	-19.4	-23.3	-28.4	-31.8	-19.4	-20.2	-21	-21.8
	90	-20	-25.3	-27.6	-28.3	-20	-28.1	-33.4	-35.8	-20	-28.6	-34.3	-38.2	-20	-22.4	-23.8	-25.3
	120	-20.4	-26.7	-29.8	-30.7	-20.4	-30.4	-36.4	-38.9	-20.4	-32.9	-39.5	-43.5	-20.4	-25	-27.3	-29.1
Sum		-23.49				-26.48				-27.61				-21.85			
Sum		-24.86															

Table D.97: Percental change in process time for different lambda assuming feedback probability 0.5.

		Sequential processes															
		Lambda=0.9				Lambda=0.7				Lambda=0.3				Lambda=0.1			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*				*Number Feedbacks*			
		0	20	40	60	0	20	40	60	0	20	40	60	0	20	40	60
Number Feed-forwards	30	-20.9	-22.3	-19.4	-17.6	-20.9	-22.7	-20.9	-20.4	-20.9	-22.2	-22.4	-22.9	-20.9	-21.1	-20.8	-20.6
	60	-20.9	-23.8	-27.2	-27.9	-20.9	-26	-30.9	-33	-20.9	-24.1	-26	-27.8	-20.9	-20.9	-20.8	-21
	90	-20.9	-27.4	-30.1	-30.9	-20.9	-30.6	-35.7	-38.1	-20.9	-28	-31.7	-34.6	-20.9	-22.2	-22.9	-23.7
	120	-20.9	-27.9	-31.6	-32.9	-20.9	-31.9	-38.1	-40.9	-20.9	-32.1	-37.5	-41.4	-20.9	-24.4	-26.1	-27.6
Sum		-25.16				-28.3				-27.14				-22.31			
Sum		-25.73															

Table D.98: Percental change in process time for different lambda assuming feedback probability 0.5.

		Mixed processes															
		Lambda=0.9				Lambda=0.7				Lambda=0.3				Lambda=0.1			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*				*Number Feedbacks*			
		0	20	40	60	0	20	40	60	0	20	40	60	0	20	40	60
Number Feed-forwards	30	-18.6	-23	-18.1	-16.8	-18.6	-22.9	-18.6	-18.4	-18.6	-21.9	-19.7	-20.8	-18.6	-19.6	-18.6	-18.7
	60	-19.9	-21.7	-24.7	-25.7	-19.9	-23.3	-28.5	-30.9	-19.9	-23.3	-27.7	-30.5	-19.9	-20.4	-20.9	-21.5
	90	-20.3	-25.9	-28.1	-28.8	-20.3	-28.8	-33.9	-36.3	-20.3	-28.5	-33.5	-37.2	-20.3	-22.4	-23.5	-24.8
	120	-20.5	-26.8	-30.2	-31.2	-20.5	-30.8	-36.8	-39.4	-20.5	-32.9	-39	-43	-20.5	-24.8	-27	-28.7
Sum		-23.77				-26.74				-27.33				-21.89			
Sum		-24.93															

Table D.99: Percental change in process time for different lambda assuming feedback probability 0.5.

		Random processes															
		Lambda=0.9				**Lambda=0.7**				**Lambda=0.3**				**Lambda=0.1**			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	30.2	144	322	413	30.2	106	206	254	30.2	60.1	85.8	97.9	30.2	45.8	54.8	59.1
	60	37	326	485	604	37	210	291	351	37	86.5	104	117	37	56.5	63.5	68.9
	90	39.2	413	600	748	39.2	258	349	421	93.2	98.2	115	129	39.2	61.4	68.6	74.7
	120	40.9	478	695	861	40.9	294	400	480	40.9	106	125	139	40.9	64.2	72.4	79.4
Sum		389.7				235.5				91.6				57.3			
Sum		193.5															

Table D.100: Cost for overlapping events for different lambda assuming feedback probability 0.5.

		Sequential processes															
		Lambda=0.9				**Lambda=0.7**				**Lambda=0.3**				**Lambda=0.1**			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	42.1	201	341	418	42.1	148	232	280	42.1	86.9	116	139	42.1	66.2	82.8	97.7
	60	42.1	374	520	635	42.1	238	318	379	42.1	100	124	145	42.1	66.2	78.9	90.3
	90	42.1	444	627	776	42.1	275	369	444	42.1	107	130	149	42.1	67	78.6	88.8
	120	42.1	505	714	883	42.1	309	414	497	42.1	113	135	153	42.1	68.3	79.2	88.6
Sum		412.9				254.4				104.1				70			
Sum		210.4															

Table D.101: Cost for overlapping events for different lambda assuming feedback probability 0.5.

		Mixed processes															
		Lambda=0.9				**Lambda=0.7**				**Lambda=0.3**				**Lambda=0.1**			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	33.1	146	368	395	33.1	109	203	248	33.1	64.2	89	101	33.1	49.5	58.9	63.9
	60	38.9	341	491	608	38.9	219	297	355	38.9	91.2	109	123	38.9	59.5	67.4	74.1
	90	40.3	421	607	756	40.3	262	354	427	40.3	100	119	134	40.3	62.5	71.3	78.4
	120	40.9	481	705	865	40.9	299	402	484	40.9	108	127	142	40.9	64.9	73.9	81.3
Sum		396.1				238.3				91.29				59.93			
Sum		196.4															

Table D.102: Cost for overlapping events for different lambda assuming feedback probability 0.5.

		Random processes															
		Lambda=0.9				Lambda=0.7				Lambda=0.3				Lambda=0.1			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	0	72.1	210	293	0	69.7	209	295	0	58.6	185.8	293	0	42	168	268
	60	0	207	341	446	0	204	340	446	0	183	328	444	0	152	300	421
	90	0	278	428	548	0	278	426	547	0	264	418	544	0	236	400	531
	120	0	331	500	629	0	334	500	628	0	327	498	628	0	299	483	624
Sum		267.9				267.3				254.5				245.3			
Sum		258.8															

Table D.103: Number of overlapping events for different lambda assuming feedback probability 0.5.

		Sequential processes															
		Lambda=0.9				Lambda=0.7				Lambda=0.3				Lambda=0.1			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	0	97.8	208	279	0	95	209	288	0	86	200	297	0	71.5	180	274
	60	0	239	361	462	0	234	360	465	0	209	344	462	0	182	318	435
	90	0	296	441	560	0	296	440	560	0	280	433	560	0	252	410	539
	120	0	348	508	638	0	350	509	638	0	340	509	643	0	313	489	631
Sum		277.5				277.8				272.7				255.9			
Sum		270.9															

Table D.104: Number of overlapping events for different lambda assuming feedback probability 0.5.

		Mixed processes															
		Lambda=0.9				Lambda=0.7				Lambda=0.3				Lambda=0.1			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	0	70.9	209	285	0	68.7	197	276	0	60	187.8	278	0	44	162	255
	60	0	207	341	446	0	214	343	446	0	194	333	447	0	164	304	424
	90	0	283	435	558	0	278	430	547	0	264	418	544	0	241	402	536
	120	0	331	500	635	0	340	500	638	0	318	507	638	0	301	487	626
Sum		268.8				267.4				261.8				246.6			
Sum		261.2															

Table D.105: Number of overlapping events for different lambda assuming feedback probability 0.5.

		Random processes											
		Feedback probability 0.1				**Feedback probability 0.3**				**Feedback probability 0.5**			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	-1.20	-1.24	-1.26	-1.29	-1.20	-1.28	-1.38	-1.56	-1.20	-1.29	-1.20	-1.22
	60	-1.05	-1.08	-1.11	-1.15	-1.05	-1.25	-1.57	-1.44	-1.05	-1.18	-1.27	-1.40
	90	-1.01	-1.05	-1.11	-1.19	-1.01	-1.46	-1.46	-1.49	-1.01	-1.24	-1.57	-1.70
	120	-1.00	-1.06	-1.15	-1.28	-1.00	-1.37	-1.41	-1.58	-1.00	-1.32	-1.82	-2.03
Average		-1.139				-1.344				-1.344			
Average		-1.276											

Table D.106: ψ values for the base case of P_4 assuming random processes.

		Sequential processes											
		Feedback probability 0.1				**Feedback probability 0.3**				**Feedback probability 0.5**			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	-1	-0.96	-0.93	-0.91	-1	-0.90	-0.90	-1.03	-1	-0.96	-1.08	-1.12
	60	-1	-0.98	-0.97	-0.97	-1	-1.05	-1.38	-1.41	-1	-1.12	-1.27	-1.41
	90	-1	-1	-1.01	-1.07	-1	-1.33	-1.37	-1.46	-1	-1.22	-1.54	-1.73
	120	-1	-1.02	-1.08	-1.19	-1	-1.29	-1.38	-1.60	-1	-1.31	-1.78	-2.05
Average		-1.00				-1.194				-1.287			
Average		-1.160											

Table D.107: ψ values for the base case of P_4 assuming sequential processes.

		Mixed processes											
		Feedback probability 0.1				**Feedback probability 0.3**				**Feedback probability 0.5**			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	-1.14	-1.14	-1.16	-1.17	-1.14	-1.16	-1.23	-1.43	-1.14	-1.20	-1.15	-1.15
	60	-1.03	-1.04	-1.05	-1.08	-1.03	-1.20	-1.52	-1.43	-1.03	-1.16	-1.27	-1.40
	90	-1.02	-1.04	-1.08	-1.15	-1.02	-1.43	-1.43	-1.48	-1.02	-1.23	-1.54	-1.70
	120	-1.01	-1.06	-1.13	-1.27	-1.01	-1.37	-1.41	-1.59	-1.01	-1.34	-1.83	-2.05
Average		-1.098				-1.305				-1.326			
Average		-1.243											

Table D.108: ψ values for the base case of P_4 assuming mixed processes.

		Random processes											
		Feedback probability 0.1				**Feedback probability 0.3**				**Feedback probability 0.5**			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	-1.20	-1.20	-1.18	-1.16	-1.20	-1.20	-1.33	-1.66	-1.20	-1.35	-0.92	-0.90
	60	-1.05	-1.02	-0.99	-0.99	-1.05	-1.27	-1.34	-0.89	-1.05	-0.99	-0.91	-0.90
	90	-1.01	-0.97	-0.96	-0.97	-1.01	-1.86	-0.98	-0.88	-1.01	-1.02	-0.98	-0.93
	120	-1.00	-0.96	-0.96	-1.00	-1.00	-1.76	-1.00	-0.95	-1.00	-1.09	-1.03	-0.97
Average		-1.039				-1.211				-1.016			
Average		-1.089											

Table D.109: ψ values for the base case of P_5 assuming random processes.

		Sequential processes											
		Feedback probability 0.1				**Feedback probability 0.3**				**Feedback probability 0.5**			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	-1	-0.95	-0.91	-0.89	-1	-0.89	-0.94	-1.09	-1	-1.00	-0.90	-0.87
	60	-1	-0.95	-0.92	-0.89	-1	-1.17	-1.18	-0.91	-1	-0.97	-0.93	-0.91
	90	-1	-0.95	-0.92	-0.91	-1	-1.65	-1.00	-0.92	-1	-1.03	-0.98	-0.94
	120	-1	-0.95	-0.93	-0.97	-1	-1.63	-1.02	-0.97	-1	-1.10	-1.03	-0.98
Average		-0.946				-1.086				-0.978			
Average		-1.033											

Table D.110: ψ values for the base case of P_5 assuming sequential processes.

		Mixed processes											
		Feedback probability 0.1				**Feedback probability 0.3**				**Feedback probability 0.5**			
		Number Feedbacks				*Number Feedbacks*				*Number Feedbacks*			
		0	**20**	**40**	**60**	**0**	**20**	**40**	**60**	**0**	**20**	**40**	**60**
Number Feed-forwards	**30**	-1.14	-1.12	-1.10	-1.08	-1.14	-1.12	-1.25	-1.54	-1.14	-1.26	-0.94	-0.90
	60	-1.03	-0.99	-0.96	-0.95	-1.03	-1.26	-1.31	-0.90	-1.03	-0.97	-0.92	-0.91
	90	-1.02	-0.97	-0.95	-0.95	-1.02	-1.81	-0.99	-0.89	-1.02	-1.03	-0.97	-0.93
	120	-1.01	-0.97	-0.95	-1.01	-1.01	-1.69	-1.01	-0.95	-1.01	-1.11	-1.04	-0.98
Average		-1.013				-1.183				-1.01			
Average		-1.068											

Table D.111: ψ values for the base case of P_5 assuming mixed processes.

E: Additional data for the UCAV process case study

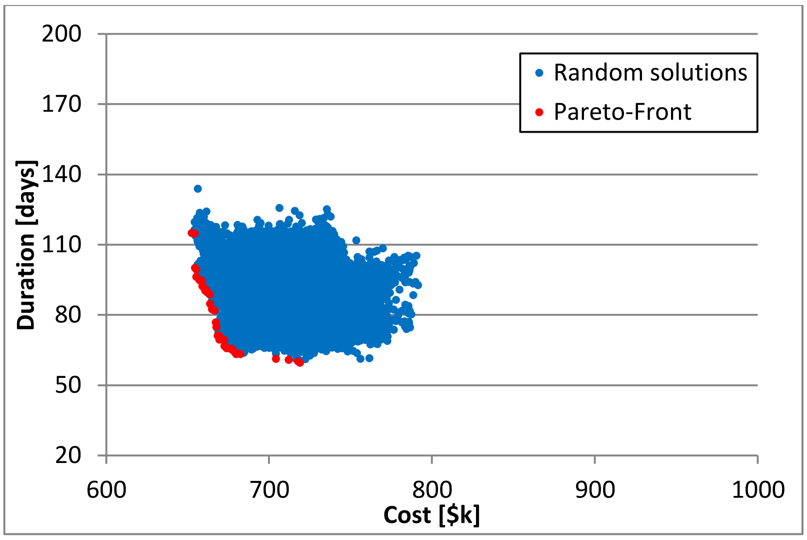


Figure E.10: Pareto-front and random solutions for the UCAV process assuming work policy P_5 and neither crashing nor overlapping. Thus, only effects of different process architectures are illustrated.

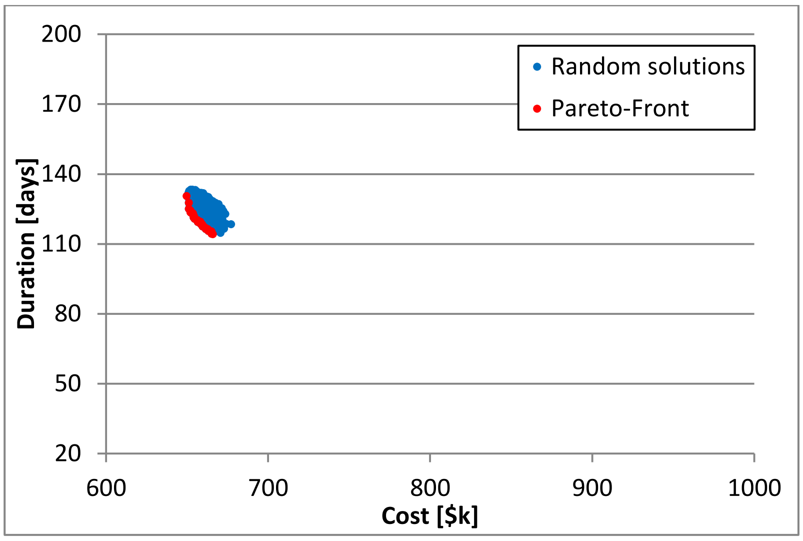


Figure E.11: Pareto-front and random solutions for the UCAV process assuming the original process architecture and work policy P_5 but no crashing. Thus, only effects of overlapping are illustrated.

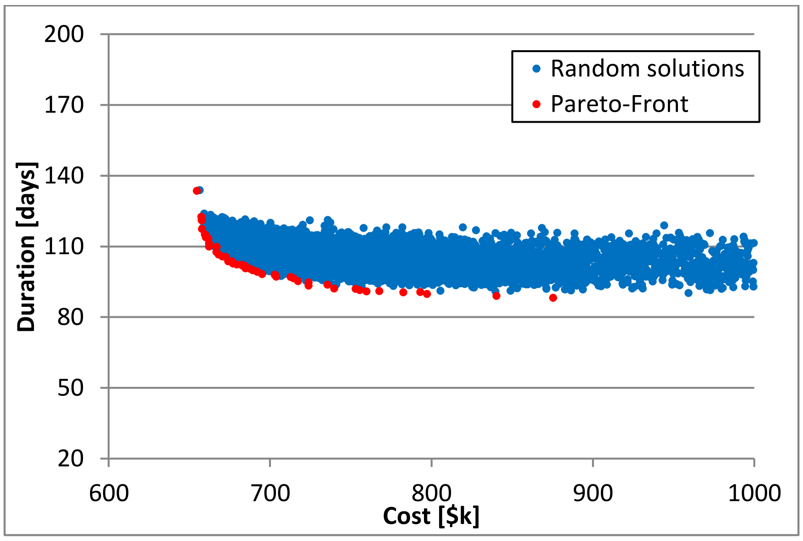


Figure E.12: Pareto-front and random solutions for the UCAV process assuming the original process architecture and work policy P_5 but no overlapping. Thus, only effects of crashing are illustrated.

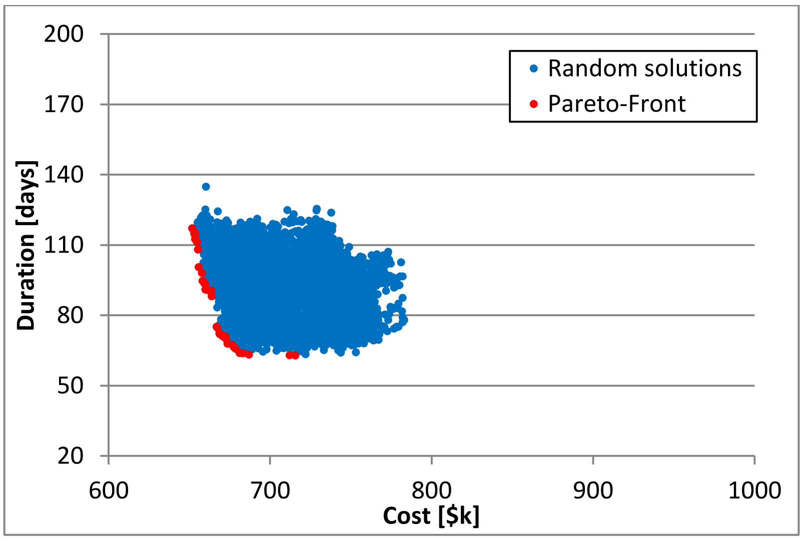


Figure E.13: Pareto-front and random solutions for the UCAV process assuming work policy P_4 and neither crashing nor overlapping. Thus, only effects of different process architectures are illustrated.

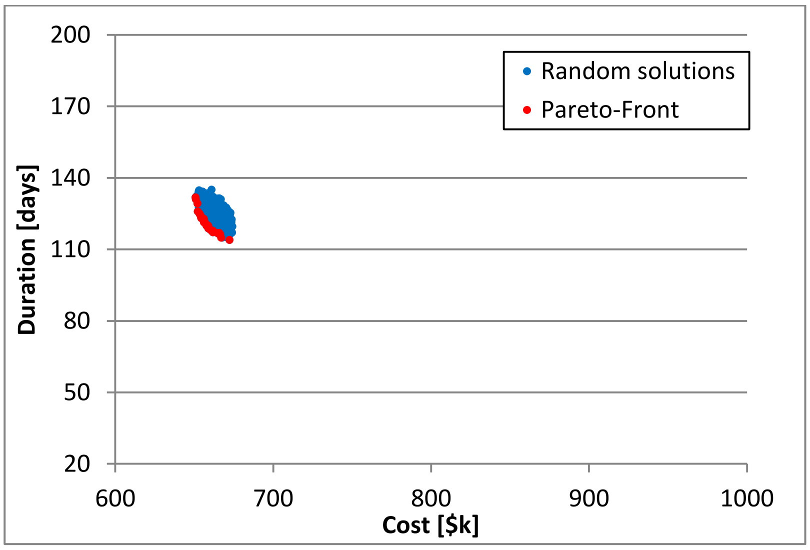


Figure E.14: Pareto-front and random solutions for the UCAV process assuming the original process architecture and work policy P_4 but no crashing. Thus, only effects of overlapping are illustrated.

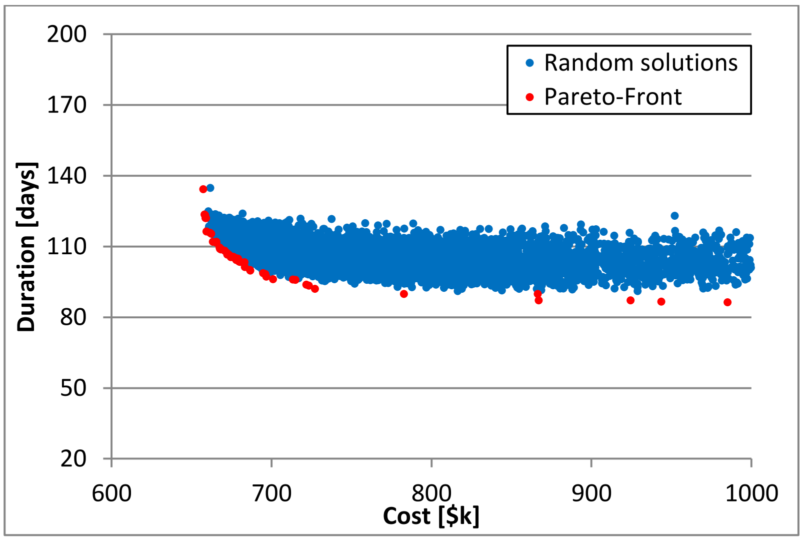


Figure E.15: Pareto-front and random solutions for the UCAV process assuming the original process architecture and work policy P_4 but no overlapping. Thus, only effects of crashing are illustrated.

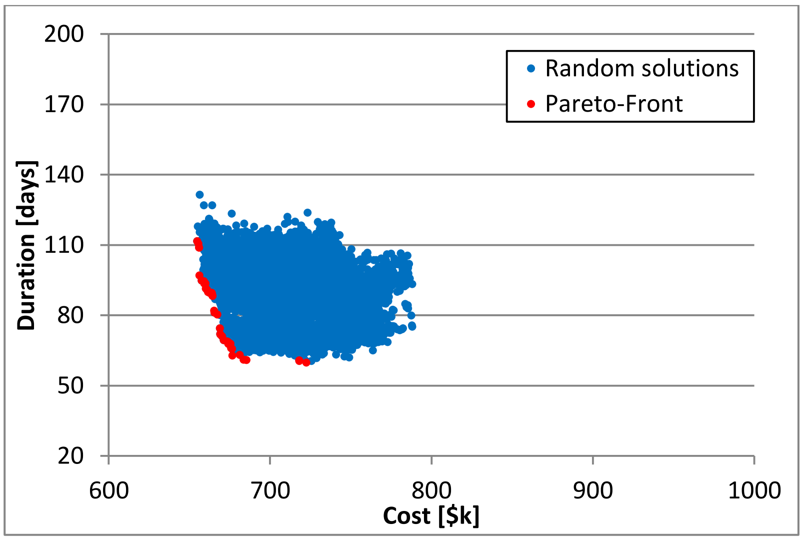


Figure E.16: Pareto-front and random solutions for the UCAV process assuming the original process architecture and work policy P_5 but no overlapping. Thus, only effects of crashing are illustrated.

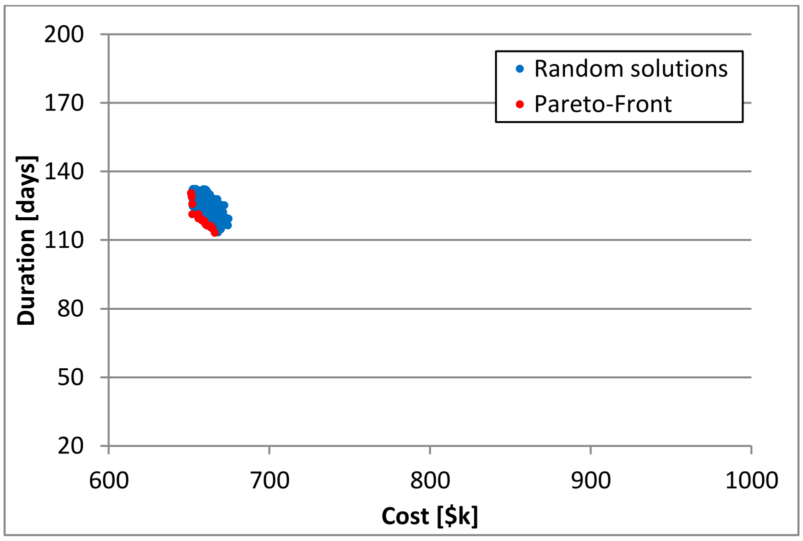


Figure E.17: Pareto-front and random solutions for the UCAV process assuming the original process architecture and work policy P_5 but no crashing. Thus, only effects of overlapping are illustrated.

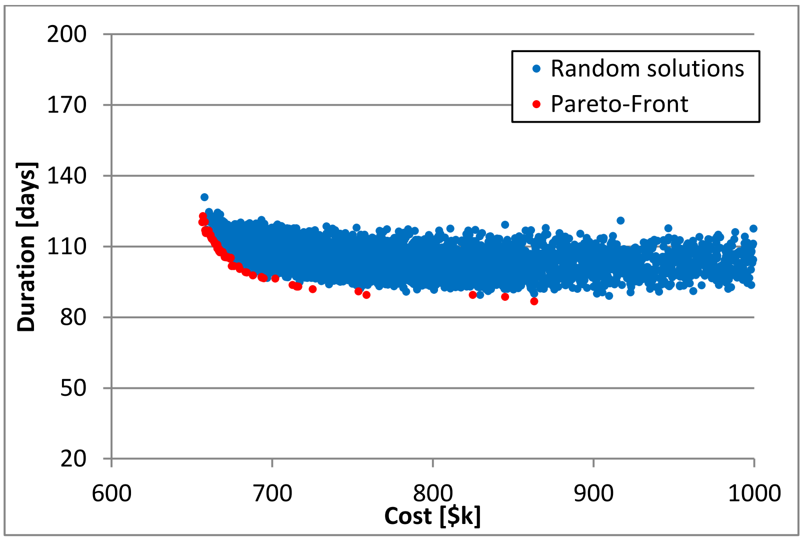


Figure E.18: Pareto-front and random solutions for the UCAV process assuming the original process architecture and work policy P_5 but no overlapping. Thus, only effects of crashing are illustrated.

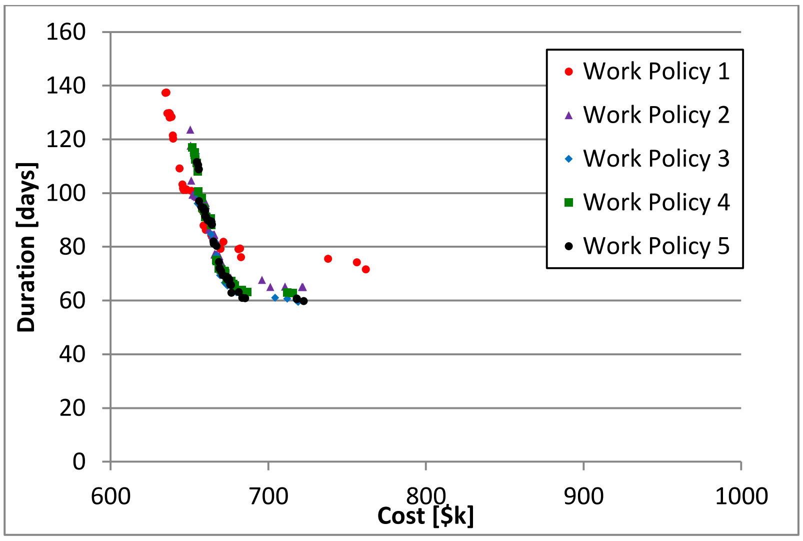


Figure E.19: Comparison of Pareto-fronts for the UCAV process assuming work policies P_1-P_5 without crashing and overlapping.

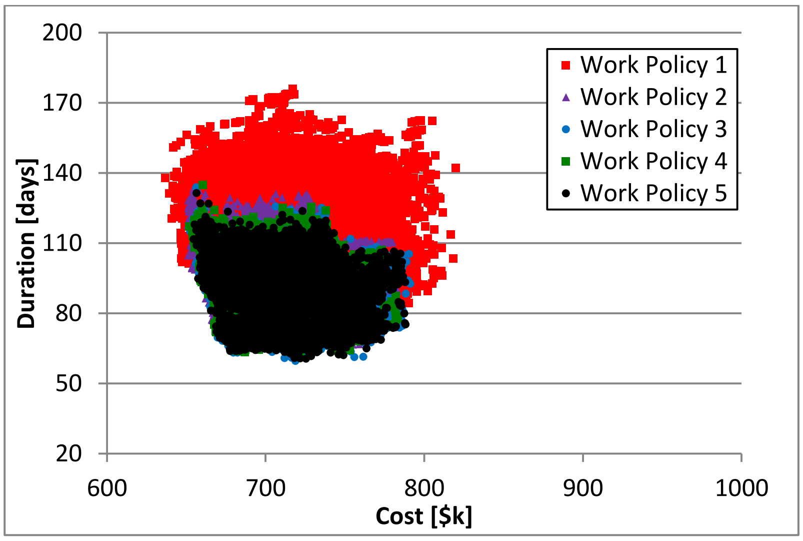


Figure E.20: Comparison of random solutions for the UCAV process assuming work policies P_1-P_5 without crashing and overlapping.

F: Additional data for the hood development process

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43
1	■																																										
2		■																																									
3			■					.6	.6						.6		.6	.6						.6																			
4		.6		■	.6	.6				.6																																	
5				.6	■	.6	.6																																				
6	.2			.2	.2	■	.2																																				
7	.2				1	1	■			.3		.3	.3			.3	.5			.3	.3			.3																			
8						1	1	■					1				1			1	1																						
9	1		1						■				1				1																										
10	.3		.3			.3	.3			■			.3				.3			.3	.3			.3																			
11						1	1				■						1				1			1																			
12			.3				.3				.3	■									.3																						
13	.3					.3							■			.3				.3																							
14						.3	.3	.3		.3			.3	■			.3			.3																							
15	.3		.3			.3	.3	.3		.3			.3	.3	■		.3	.3		.3																							
16						.6							.6			■				.6																							
17	1					1								1		1	■			1																							
18																	1	■																									
19	.6		.6				.6			.6			.6						■																								
20			1														1			■																							
21			1				1									1	1			1	■	1	1																				
22			.6			.6											.6			.6	.6	■	.6																				
23	.6					.6	.6	.6		.6			.6	.6			.6			.6	.6	.6	■			.6	.6	.6															
24			.3				.3			.3			.3	.3		.3	.3		.3	.3	.3	.3	.3	■																			
25										.3			.3	.3			.3			.3	.3	.3	.3	.3	■				.3	.3	.3												
26						.1															.1		.1	.1	.1	■			.1	.1	.1												
27						.6															.6	.6	.6		.6		■	.6		.6	.6												
28						.6															.6		.6		.6		.6	■		.6	.6												
29																				.3		.3	.3		.3				■														
30																							.3						.3	■	.3				.3							.3	
31	.6																						.6		.6				.6		■											.6	
32																						.6	.6		.6				.6	.6	.6	■											
33																																1	■	1									
34																																	1	■									
35																			.6														1	1	■								
36																																1	1	1		■							
37																																		1	1	1	■						
38																																	1					■					
39																																			1	1	1	1	■	1	1	1	
40																													1	1	1				1	1	1	1	1	■			
41																																							1	1	■		
42																																			1	1	1	1	1			■	
43																																		1	1	1	1	1	1	1	1	1	■

Figure F.21: DSM M_2 for the hood development process.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43
1	■																																										
2		■																																									
3			■					1	1						1		1	1						1																			
4		1		■	1	1				1																																	
5				1	■	1	1																																				
6	1			1	1	■	1																																				
7	1				1	1	■			1		1	1			1	1			1	1			1																			
8						1	1	■					1				1			1	1																						
9	1		1						■				1				1																										
10	1		1			1	1			■			1				1			1	1			1																			
11						1	1				■						1				1			1																			
12			1				1				1	■									1																						
13	1					1							■			1				1																							
14						1	1	1		1			1	■			1			1																							
15	1		1			1	1	1		1			1	1	■		1	1		1																							
16						1							1			■				1																							
17	1					1								1		1	■			1																							
18																	1	■																									
19	1		1				1			1			1						■																								
20			1														1			■																							
21			1				1									1	1			1	■	1	1																				
22			1			1											1			1	1	■	1																				
23	1					1	1	1		1			1	1			1			1	1	1	■			1	1	1															
24			1				1			1			1	1		1	1		1	1	1	1	1	■																			
25										1			1	1			1			1	1	1	1	1	■				1	1	1												
26						.1															1		1	1	.1	■			1	1	1												
27						1															1	1	1		1		■	1		1	1												
28						1															1		1		1		1	■		1	1												
29																				1		1	1		1				■														
30																							1						1	■	1				1							1	
31	1																						1		1				1		■											1	
32																						1	1		1				1	1	1	■											
33																																1	■	1									
34																																	1	■									
35																			1														1	1	■								
36																																1	1	1		■							
37																																		1	1	1	■						
38																																	1					■					
39																																			1	1	1	1	■	1	1	1	
40																													1	1	1				1	1	1	1	1	■			
41																																							1	1	■		
42																																			1	1	1	1	1			■	
43																																		1	1	1	1	1	1	1	1	1	■

Figure F.22: DSM M_3 for the hood development process.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43
1	■																																										
2		■																																									
3			■					0	0						0		0	0						0																			
4		0		■	0	0				0																																	
5				0	■	0	0																																				
6	0			0	0	■	0																																				
7	0				0	0	■			0		0	0			0	0			0	0			0																			
8						0	0	■					0				0			0	0																						
9	0		0						■				0				0																										
10	0		0			0	0			■			0				0			0	0			0																			
11						0	0				■						0				0			0																			
12			0				0				0	■									0																						
13	0					0							■			0				0																							
14						0	0	0		0			0	■			0			0																							
15	0		0			0	0	0		0			0	0	■		0	0		0																							
16						0							0			■				0																							
17	0					0								0		0	■			0																							
18																	0	■																									
19	0		0				0			0			0						■																								
20			0														0			■																							
21			0				0									0	0			0	■	0	0																				
22			0			0											0			0	0	■	0																				
23	0					0	0	0		0			0	0			0			0	0	0	■			0	0	0															
24			0				0			0			0	0		0	0		0	0	0	0	0	■																			
25										0			0	0			0			0	0	0	0	0	■				0	0	0												
26						.0															0		0	0	0	■			0	0	0												
27						0															0	0	0		0		■	0		0	0												
28						0															0		0		0		0	■		0	0												
29																				0		0	0		0				■														
30																							0						0	■	0				0							0	
31	0																						0		0				0		■											0	
32																						0	0		0				0	0	0	■											
33																																0	■	0									
34																																	0	■									
35																			0														0	0	■								
36																																0	0	0		■							
37																																		0	0	0	■						
38																																	0					■					
39																																			0	0	0	0	■	0	0	0	
40																													0	0	0				0	0	0	0	0	■			
41																																							0	0	■		
42																																			0	0	0	0	0			■	
43																																		0	0	0	0	0	0	0	0	0	■

Figure F.23: DSM M_4 for the hood development process.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43
1	■																																										
2		■																																									
3			■					.9	.9						.9		.9	.8						.9																			
4		1		■	1	.9				.8																																	
5				1	■	.9	1																																				
6	1			1	1	■	1																																				
7	1				1	.9	■			.7		.8	.9			.8	.9			.9	.9			.9																			
8						.8	1	■					.9				.9			.9	.9																						
9	1		.8						■				.9				.9																										
10	1		.8			.8	1			■			.9				.9			.9	.9			.9																			
11						.8	1				■						.9				.9			.9																			
12			.7				1				.9	■									.9																						
13	1					.8							■			.8				.9																							
14						.8	1	.7		.8			1	■			.9			.9																							
15	1		.8			.8	1	.7		.7			1	1	■		.9	.8		.9																							
16						.8							1			■				.9																							
17	1					.8								1		1	■			.9																							
18																	1	■																									
19	1		.8				1			.7			1						■																								
20			.8														1			■																							
21			.8				1									1	1			1	■	.9	.9																				
22			.8			.8											1			1	1	■	.9																				
23	1					.8	1	.9		.9			1	1			1			1	1	1	■			.9	.9	.9															
24			.8				1			.9			1	1		1	1		1	1	1	1	1	■																			
25										.9			1	1			1			1	1	1	1	1	■				.9	.9	.9												
26						.8															.1		.1	.1	.1	■			.9	.9	.9												
27						.8															1	1	1		1		■	.9		.9	.9												
28						.8															1		1		1		1	■		.9	.9												
29																				1		1	1		1				■														
30																							1						1	■	.9				.9							.9	
31	1																						1		1				1		■											.9	
32																						1	1		1				1	1	1	■											
33																																1	■	.9									
34																																	1	■									
35																			1														1	1	■								
36																																1	1	1		■							
37																																		1	1	1	■						
38																																	1					■					
39																																			1	1	1	1	■	.9	.9	.9	
40																													1	1	1				1	1	1	1	1	■			
41																																							1	1	■		
42																																			1	1	1	1	1			■	
43																																		1	1	1	1	1	1	1	1	1	■

Figure F.24: DSM M_5 for the hood development process.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43
1	■																																										
2		■																																									
3			■					.2	.2						.1		.1	.2						.1																			
4		0		■	0	.3				.4																																	
5				0	■	.3	0																																				
6	0			0	0	■	0																																				
7	0				0	.1	■			.4		.3	.1			.3	.1			.1	.1			.1																			
8						.1	0	■					.1				.1			.1	.1																						
9	0		.1						■				.1				.1																										
10	0		.3			.1	0			■			.1				.1			.1	.1			.1																			
11						.1	0				■						.1				.1			.1																			
12			.3				0				.2	■									.1																						
13	0					.2							■			.3				.1																							
14						.3	0	.3		.3			0	■			.1			.1																							
15	0		.5			.5	0	.5		.5			0	0	■		.1	.2		.1																							
16						.2							0			■				.1																							
17	0					.2								0		0	■			.1																							
18																	0	■																									
19	0		.5				0			.5			0						■																								
20			.5														0			■																							
21			.5				0									0	0			0	■	.2	.2																				
22			.5			.3											0			0	0	■	.2																				
23	0					.1	0	.1		.1			0	0			0			0	0	0	■			.1	.1	.1															
24			.5				0			.1			0	0		0	0		0	0	0	0	0	■																			
25										.1			0	0			0			0	0	0	0	0	■				.2	.2	.2												
26						.1															0		0	0	0	■			.2	.2	.2												
27						.1															0	0	0		0		■	.1		.2	.2												
28						.1															0		0		0		0	■		.2	.2												
29																				0		0	0		0				■														
30																							0						0	■	0				.2							.2	
31	0																						0		0				0		■											.2	
32																						0	0		0				0	0	0	■										
33																																0	■	.1									
34																																	0	■									
35																			0														0	0	■								
36																																0	0	0		■							
37																																		0	0	0	■						
38																																	0					■					
39																																			0	0	0	0	■	.2	.2	.2	
40																													0	0	0				0	0	0	0	0	■			
41																																							0	0	■		
42																																			0	0	0	0	0			■	
43																																		0	0	0	0	0	0	0	0	0	■

Figure F.25: DSM M_6 for the hood development process.

Activity Name	Continuous or Discrete?	Intensity (α)	Maximum Crashing	Resource
Strategies for product, mkt, mfg, supply, design, and reusability confirmed (Est. PDL)	C	-	0%	-
Select powertrain lineup	C	-	0%	-
Select materials for all system components	C	Medium = 6	25%	Staff
Freeze proportions and selected hardpoints	C	Medium = 6	25%	Staff
Verify that hardpoints and structural joint designs are compatible w/ program targets	C	High = 12	10%	Staff
Approve master sections	C	High = 12	10%	Staff
Develop initial design concept (preliminary CAD model)	C	High = 12	30%	Staff
Estimate blank size	C	Medium = 6	10%	Staff
Estimate efforts	C	Medium = 6	10%	Staff
Develop initial attachment scheme	C	High = 12	25%	Staff
Estimate latch loads	C	Medium = 6	10%	Staff
Cheat outer panel surface	C	Medium = 6	30%	Staff
Define hinge concept	C	High = 12	10%	Staff
Get prelim. mfg and asy feas. (form, holes, hem, weld patterns, mastic locations, adhesive)	C	High = 12	15%	Staff
Perform cost analysis (variable and investment)	C	High = 12	20%	Staff
Perform swing study	C	High = 12	20%	Staff
Theme approval for interior and exterior appearance (prelim surf available)	C	High = 12	10%	Staff
Marketing commits to net revenue; initial ordering guide available	C	Medium = 6	10%	Staff
Program DVPs and FMEAs complete	C	Medium = 6	5%	Staff
Approved theme refined for craftsmanship execution (consistent w/ PA objectives)	C	High = 12	10%	Staff
PDN0 - Interior and exterior Class 1A surfaces transferred to engineering (+/- 01mm)	C	High = 12	20%	Staff
Conduct cube review and get surface buyoff	C	High = 12	10%	Staff
Verify mfg and asy feas. (form, holes, hem, weld patterns, mastic locations, adhesive)	C	High = 12	15%	Staff
Evaluate functional performance (analytically)	C	High = 12	15%	Staff
PDN 1 - Release system design intent level concept to manufacturing	C	High = 12	20%	Staff
Develop stamping tooling	C	High = 16	25%	Machinery
Develop hemming tooling (if applicable)	C	High = 16	25%	Machinery
Develop assembly tooling	C	High = 16	25%	Machinery
PDN2 - Last Class 1surface verified and released for major formed parts	C	High = 12	20%	Staff
PDN3 - Final math 1, 2, & 3 data released	C	High = 12	20%	Staff
CAD files reflect pre-CP verification changes	C	Medium = 6	5%	Staff
Make "like production" part and asy tools / ergonomics / process sheets (to extent feasible)	C	High = 12	10%	Staff
First CPs available for tuning and durability testing	C	High = 6	10%	Staff
Complete CMM analysis of all end items & subassemblies	C	High = 16	10%	Staff
Perform DV tests (physical)	C	High = 12	20%	Staff
Verify manufacturing and assembly process capability	C	High = 16	15%	Staff
Complete prelim. ESO for: CP durability testing	C	Medium = 6	10%	Staff
Complete prelim. ESO for: Initial set of road tests completed	C	Medium = 6	10%	Staff
Complete prelim. ESO for: Known changes from CP containable for 1PP	C	Medium = 6	10%	Staff
Complete prelim. ESO for: Design is J1 level - no further changes except No-Blds	C	Medium = 6	10%	Staff
Supplier commitment to support 1PP w/ PSW parts	C	Medium = 6	10%	Staff
Complete prelim. ESO for: Eng. confidence that objectives will be met declared	C	Medium = 6	10%	Staff
Readiness to proceed to tool tryout (TTO), 1PP and Job #1	C	Medium = 6	10%	Staff

Table F.112: Crashing related data for the hood development process.

List of Figures

List of Tables

List of Definitions

Bibliography

Abdelsalam, H.M.E. and H.P. Bao (2006) "A Simulation-based Optimization Framework for Product Development Cycle Time Reduction," *IEEE Transactions on Engineering Management,* **53**(1): 69-85.

Adlakha, V.G. (1986) "An improved conditional Monte Carlo technique for the stochastic shortest path problem," *Management Science,* **32**(10), 1360-1367.

Adler, P.S., A. Mandelbaum, V. Nguyen and E. Schwerer (1995) "From Project to Process Management: An Empirically-based Framework for Analyzing Product Development Time," *Management Science,* **41**(3), 458-484.

Ahmadi, R.H., T.A. Roemer and R.H. Wang (2001) "Structuring Product Development Processes," *European Journal of Operational Research,* **130**(3): 539-558.

AitSahlia, F., E. Johnson and P. Will (1995) "Is Concurrent Engineering Always a Sensible Proposition?" *IEEE Transactions on Engineering Management,* **42**(2): 166-170.

Bäck, T., U. Hammel and H.-P. Schwefel (1997) "Evolutionary computation: Comments on the history and current state," *IEEE Transactions on Evolutionary Computation*, **1**(1): 3–17.

Bean, J.C. (1994) "Genetic Algorithms and Random Keys for Sequencing and Optimization," *Journal on Computing.* **6**(2): 154–160.

Berman, E.B. (1964) "Resource allocation in a PERT network under continuous activity time-cost functions," *Management Science.* **10**(4): 734–745.

Bhuiyan, N. (2004) "Simulation of the New Product Development Process for Performance Improvement," *Management Science*, **50**(12), 1690-1703.

Brooks, F.P. Jr. (1995) *The Mythical Man-Month: Essays on Software Engineering,* Anniversary Edition, Reading, MA: Addison-Wesley.

Browning, T.R. (1998) *Modeling and Analyzing Cost, Schedule, and Performance in Complex System Product Development*, Ph.D. Thesis, Massachusetts Institute of Technology, Cambridge, MA.

Browning, T.R. (2001) "Applying the Design Structure Matrix to System Decomposition and Integration Problems: A Review and New Directions," *IEEE Transactions on Engineering Management*, **48**(3): 292-306.

Browning, T.R. and S.D. Eppinger (2002) "Modeling Impacts of Process Architecture on Cost and Schedule Risk in Product Development," *IEEE Transactions on Engineering Management,* **49**(4): 428-442.

Browning, T.R. and R.V. Ramasesh (2007), "A Survey of Activity Network-Based Process Models for Managing Product Development Projects," *Production and Operations Management,* **16**(2), 217-240.

Calantone, R.J. and C.A. Di Benedetto (2000) "Performance and Time to Market: Accelerating Cycle Time with Overlapping Stages," *IEEE Transactions on Engineering Management,* **47**(2): 232-244.

Chen, J., R.R. Reilly and G.S. Lynn (2005) "The impacts of Speed-to-Market on New Product Success: The Moderating Effects of Uncertainty," *IEEE Transactions on Engineering Management*, **52**(2), 2005.

Cho, S.-H. and S.D. Eppinger (2005) "A Simulation-Based Process Model for Managing Complex Design Projects," *IEEE Transactions on Engineering Management,* **52**(3): 316-328.

Clark, Kim B. and Takahiro Fujimoto (1991) *Product Development Performance: Strategy, Organization, and Management in the World Auto Industry*, Boston: Harvard Business School Press.

Coello, C.A. Coello, G.T. Pulido and M.S. Lechuga (2004) "Handling Multiple Objectives With Particle Swarm Optimization," *IEEE Transactions on Evolutionary Computation,* **8**(3): 256-279.

Coello, C.A. Coello, D.A. Van Veldhuizen and G.B. Lamont (2007) *Evolutionary Algorithms for Solving Multi-objective Problems*, Norwell, MA: Kluwer.

Collette, Y. and P. Siarry (2003) *Multiobjective Optimization*, Springer.

Coverstone-Carroll, V., J.W. Hartmann and W.J. Mason (2000) "Optimal multi-objective low-thrust spacecraft trajectories," *Computer Methods in Applied Mechanics and Engineering*, **186**(2-4): 387-402.

De Weck, O. (2006) System Project Management. Course ESD 36, Massachusetts Institute of Technology.

Deb, K. (1999) "Multi-objective genetic algorithms: Problem difficulties and construction of test problems," *Evolutionary Computation*,**7**(3): 205-230.

Deb, K. (2009) *Multi-Objective Optimization Using Evolutionary Algorithms*, John Wiley & Sons, Chichester, UK.

Deb, K., M. Mohan and S. Mishra (2003) "Towards a quick computation of well-spread pareto-optimal solutions," *Evolutionary Multi-Criterion Optimization. Second International Conference, EMO 2003,* pp. 222-236.

Deb, K., M. Mohan and S. Mishra (2005) "Evaluating the ε-Domination Based Multi-Objective Evolutionary Algorithm for a Quick Computation of Pareto-Optimal Solutions", *Evolutionary Computation*, **13**(4): 501-525.

Deb, K., A. Pratap, S. Agarwal, and T. Meyarivan (2002) "A fast and elitist multiobjective genetic algorithm:Nsga-II," *IEEE Transactions On Evolutionary Computation*, **6**(2): 182-197.

Doerner, K., W.J. Gutjahr, R.F. Hartl, C. Strauss and C. Stummer (2004) "Pareto ant colony optimization: A metaheuristic approach to multiobjective portfolio selection," *Annals of Operations Research*, **131**:79-99.

Eastman, R.M. (1980) "Engineering Information Release Prior to Final Design Freeze," *IEEE Transactions on Engineering Management,* **27**(2): 37-41.

Eppinger, S.D. (2001) "Innovation at the Speed of Information," *Harvard Business Review,* **79**(1): 149-158.

Eppinger, S.D., M.Nukala and D. Whitney (1997) "Generalized models of design iteration using signal flow graphs," *Research in Engineering Design*, **9**(2), 112-123.

Eppinger, S.D., D.E. Whitney, R.P. Smith and D.A. Gebala (1994) "A Model-Based Method for Organizing Tasks in Product Development," *Research in Engineering Design,* **6**(1): 1-13.

Everson, R.M. and J.E. Fieldsend (2006) "Multiobjective Optimization of Safety Related Systems: An Application to Short-Term Conflict Alert," *IEEE Transactions on Evolutionary Computation,* **10**(2): 187-198.

Falk, J.E. and J.L. Horowitz (1972) "Critical path problems with concave cost-time curves," *Management Science*. **19**(4): 446–455.

Ford, D.N. and J.D. Sterman (1998) "Dynamic Modeling of product development processes," *System Dynamics Review*, **14**(1): 31-68.

Ford, D.N. and J.D. Sterman (2003) "The Liar's Club: Concealing Rework in Concurrent Development," *Concurrent Engineering: Research and Applications*, **11**(3): 211-219.

Fonseca, C.M. and P.J. Fleming (1998) "Multiobjective optimization and multiple constraint handling with evolutionary algorithms-part1: a unified formulation," *IEEE Transactions On Systems, Man and Cybernatics-Part A: Systems and Humans*, **28**(1): 26-37.

Fricke, E., B. Gebhard, H. Negele and E. Igenbergs (2000) "Coping with changes: Causes, findings and strategies," *Systems Engineering*, **3**(4):169-179.

Fujita, K., N. Hirokawa, S. Akagi, S. Kitamura and H. Yokohata (1998) "Multi-objective optimal design of automotive engine using genetic algorithms," Proceedings of 1998 ASME Design Engineering Technical Conferences.

Fulkerson, D.R. (1961) "A Network Flow Computation for Project Cost Curves," *Management Science*, **7**(2): 167-178.

Funk, J.L. (1997) "Concurrent Engineering and the Underlying Structure of the Design Problem," *IEEE Transactions on Engineering Management*, **44**(3): 305-315.

Galbraith, C., D. Thomas and M. Finn (2003) "Manufacturing Simulation of an Automotive Hood Assembly," 4th European LS-DYNA Users Conference.

Gebala, D.A. and S.D. Eppinger (1991) "Methods for Analyzing Design Procedures," Proceedings of the ASME International Design Engineering Technical Conferences (Design Theory & Methodology Conference), pp. 227-233.

Gerk, J.E.V. and R.Y. Qassim (2008) "Project Acceleration via Activity Crashing, Overlapping, and Substitution," *IEEE Transactions on Engineering Management*, **55**(4): 590-601.

Goldberg, D.E. (1989) *Genetic algorithms in search, optimization, and machine learning*. New York, NY: Addison - Wesley.

Goldberg, D.E. and K. Deb (1991) "A comparative analysis of selection schemes used in Genetic Algorithms," *Foundations of Genetic Algorithms*.

Goldberg, D.E., K. Deb and D. Thierens (1991) "Toward a better understanding of mixing in Genetic Algorithms," *Proceedings of the 4th International Conference on Genetic Algorithms*.

Glover, F. and G.A. Kochenberger (2003) *Handbook of metaheuristics*. Norwell, MA: Kluwer.

Hab, G. and R. Wagner (2006) *Projektmanagement in der Automobilindustrie*. Gabler-Verlag.

Hanne, T. (1999) "On the convergence of multiobjective evolutionary algorithms," *European Journal Of Operational Research*, **117**(3): 553-564.

Helbig, S. and D. Pateva (1994) "On several concepts for ε-efficiency," *OR Spektrum*, **16**(3): 179–186.

Holland, J.H. (1975) *Adaptation in natural and artificial systems*. Ann Arbor, MI: The University of Michigan Press.

Iman, R.L., J.E. Campbell and J.C. Helton (1981) "An approach to sensitivity analysis of computer models. Part 1- Introduction, input, variable selection and preliminary variable assessment", *Journal of Quality Technology*, **13**: 174-183.

Iredi, S., D. Merkle and M. Middendorf (2001) "Bi-criterion optimization with multi colony ant algorithms," Proceedings First International Conference on Evolutionary Multicriterion Optimization.

Kanda, A. and U. R. K. Rao (1984) "A network flow procedure for project crashing with penalty nodes," *European. Journal of Operational Research*, **16**(2): 174–182

Kelley, J.E. (1961) "Critical-Path Planning and Scheduling: Mathematical Basis," *Operations Research*, **9**(3): 296-320.

Kelley, J. E. and M. R. Walker (1959) "Critical-path planning and scheduling," Proceedings Eastern Joint Computational Conference, pp. 160–173.

Kerzner, H. (2005) *Project management: a systems approach to planning, scheduling, and controlling*, Hoboken, NJ: Wiley.

Knjazew, D. (2002) *OmeGA: A Competent Genetic Algorithm for Solving Permutation and Scheduling Problems*, Norwell, MA: Kluwer Academic Publishers Group

Krishnan, V. and K.T. Ulrich (2001) "Product Development Decisions: A Review of the Literature," *Management Science,* **47**(1): 1-21.

Krishnan, V., S.D. Eppinger and Daniel E. Whitney (1997a) "A Model-Based Framework to Overlap Product Development Activities," *Management Science,* **43**(4): 437-451.

Krishnan, V., S.D. Eppinger and Daniel E. Whitney (1997b) "Simplifying Iterations in Cross-Functional Design Decision Making," *ASME Journal of Mechanical Design,* **119**(4): 485-493.

Kumar, R. and P. Rockett (2002) "Improved sampling of the pareto-front in multiobjective genetic optimizations by steady-state evolution: A pareto converging genetic algorithm," *Evolutionary Computation,* **10**(3): 283-314.

Kusiak, A. and J.Wang (1993) "Decomposition of the Design Process" *ASME Journal of Mechanical Design*, **115**(4): 687-695.

Kusiak, A., N. Larson and J. Wang (1994) "Reengineering of Design and Manufacturing Processes," *Computers and Industrial Engineering*, **26**(3): 521-536.

Langerak, F. and E. Hultink (2005) "The impact of new product development acceleration approaches on speed and profitability: lessons for pioneers and fast followers," *IEEE Transactions on Engineering Management*, **52**(3): 336-349.

Laumanns, M., L. Thiele, K. Deb, and E. Zitzler (2002) "Combining convergence and diversity in evolutionary multiobjective optimization," *Evolutionary Computation*, **10**(3): 263-282.

Lévárdy, V. and T.R. Browning (2009) "An Adaptive Process Model to Support Product " *IEEE Transaction on Engineering Management*, **56**(4):600-620.

Lieberman, M.B. and D.B. Montgomery (1988) "First-mover advantages," *Strategic Management Journal*, **9**(1): 41-58.

Loch, C.H. and C. Terwiesch (2005) "Rush and Be Wrong or Wait and Be Late? A Model of Information in Collaborative Processes," *Production and Operations Management,* **14**(3): 331-343.

MacCormack, A.D. and R. Verganti (2003) "Managing the sources of uncertainty: Matching process and context in software development," *Journal of Production Innovation Management,* **20**(3): 217-232.

MacCormack, A.D., R. Verganti and M. Iansiti (2001) "Developing Products on "Internet Time": The Anatomy of a Flexible Development Process," *Management Science,* **47**(1): 133-150.

Malcolm, D., J. Roseboom, C. Clark and W. Fazar (1959) "Application of a technique for research and development program evaluation," *Operations Research,* **7**(5): 646-669.

Mansfield, E., J. Rapoport, J. Schnee, S. Wagner and M. Hamburger (1971) *Research and Innovation in the Modern Corporation*, First Ed., W.W. Norton & Company, New York.

Miettinen, K. (1999) *Nonlinear Multiobjective Optimization*, Boston, MA: Kluwer.

Meier, C. (2006) *Optimization of Activity-based Design Structure Matrices Using Genetic Algorithms*, Master's Thesis (Computer Science), Technische Universität München, written at the University of Illinois Urbana-Champaign.

Meier, C., A.A. Yassine and T.R. Browning (2007) "Design Process Sequencing with Competent Genetic Algorithms," *ASME Journal of Mechanical Design,* **129**(6): 566-585.

Neumann, K. and U. Steinhardt (1979) *GERT Networks and the Time-Oriented Evaluation of Projects,* Springer-Verlag New York, NJ.

Poloni, C., A. Giurgevich, L. Onesti and V. Pediroda (2000) "Hybridization of a multiobjective genetic algorithm, a neural network and a classical optimizer for complex design problems in fluid dynamics," *Computer Methods in Applied Mechanics and Engineering,* **186**(2-4): 403-420.

Roemer, T.A. and R. Ahmadi (2004) "Concurrent Crashing and Overlapping in Product Development," *Operations Research,* **52**(4): 606-622.

Roemer, T.A., R. Ahmadi and R.H. Wang (2000) "Time-Cost Trade-Offs in Overlapped Product Development," *Operations Research,* **48**(6): 858-865.

Rudolph, G. and A. Agapie (2000) "Convergence properties of some multi-objective evolutionary algorithms," *Congress on Evolutionary Computation (CEC 2000),* pp. 1010-1016.

Safoutin, M. and R. P. Smith (1998) "Classification of Iteration in Engineering Design Processes," *Proceedings of the ASME International Design Engineering Technical Conferences (Design Theory & Methodology Conference),* Atlanta, Sep. 13-16.

Sastry, K. and D.E. Goldberg (2001) "Modeling tournament selection with replacement using apparent added noise," *Proceedings of the Genetic and Evolutionary Computation Conference*.

Schaffer, J.D. (1984) *Multiple Objective Optimization with Vector Evaluated Genetic Algorithms*. Ph.D. thesis, Vanderbilt University, Nashville, Tennessee.

Smith, K., R. Everson and J. Fieldsend (2004) "Dominance measures for multi-objective simulated annealing," *Proceedings of Congress on Evolutionary Computation*.

Smith, P.G. and D.G. Reinertsen (1997) *Developing products in half the time: new rules, new tools*. Wiley.

Smith, R.P. (1997) "The Historical Roots of Concurrent Engineering Fundamentals," *IEEE Transactions on Engineering Management*, **44**(1): 67-78.

Smith, R.P. and S.D. Eppinger (1997) "Identifying Controlling Features of Engineering Design Iteration," *Management Science,* **43**(3): 276-293.

Srinivas, N. and K. Deb (1994) "Multiobjective optimization using nondominated sorting in genetic algorithms," *Evolutionary Computation*, **2**(3): 221-248.

Standish Group International (2004) "The Chaos Report".

Steward, D.V. (1981) *Systems Analysis and Management: Structure, Strategy and Design,* Petrocelli Books, New York.

Tatikonda, Mohan V. and Mitzi M. Montoya-Weiss (2001) "Integrating Operations and Marketing Perspectives of Product Innovation: The Influence of Organizational Process Factors and Capabilities on Development Performance," *Management Science,* **47**(1): 151-172.

Terwiesch, Christian and Christoph H. Loch (1999) "Measuring the Effectiveness of Overlapping Development Activities," *Management Science,* **45**(4): 455-465.

Ulrich, K. T. and S.D. Eppinger (2004) *Product Design and Development*, 3rd Edition, New York: McGraw-Hill, Inc..

Wang, Z. and H.-S. Yan (2005) "Optimizing the Concurrency for a Group of Design Activities," *IEEE Transactions on Engineering Management*, **52**(1): 102-118.

Whitney, Daniel E. (1990) "Designing the Design Process," *Research in Engineering Design,* **2**(1): 3-13.

Wolpert, D.H. and W.G. Macready (1997) "No Free Lunch Theorems for Search," *IEEE Transactions on Evolutionary Computation*, **1**(1): 67-82.

Yassine, A A., *et al.* (2003) "Information Hiding in Product Development: The Design Churn Effect," *Research in Engineering Design*, **14**(3): 145-161.

Yassine, A.A., D. E. Whitney, J. Lavine and T. Zambito (2000) "Do-It-Right-First-Time (DRFT) Approach to DSM Restructuring," *Proceedings of the ASME International Design Engineering Technical Conferences (Design Theory & Methodology Conference)*, Baltimore.

Yassine, A.A. and D. Braha (2003) "Complex Concurrent Engineering and the Design Structure Matrix Method," *Concurrent Engineering: Research and Applications,* **11**(3): 165-176.

Zambito, T. (2000) *Using the Design Structure Matrix to Structure Automotive Hood System Development*, Masters Thesis, Massachusetts Institute of Technology, Cambridge, MA.

Zitzler, E. and L. Thiele (1999) "Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach," *IEEE Transactions on Evolutionary Computation*, **3**(4): 257-271.

Zitzler, E., K. Deb and L. Thiele (2000) "Comparison of multiobjective evolutionary algorithms: Empirical results," *Evolutionary Computation*, **8**(2): 173-195.

Zitzler, E., M. Laumanns and L. Thiele (2002) "SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization," *Evolutionary Methods for Design, Optimisation, and Control*, Barcelona, Spain, pp 19-26.