Realia et naturalia

DISSERTATIONES
MATHEMATICAE
UNIVERSITATIS
TARTUENSIS
101

FREDRIK MILANI

On Sub-Processes, Process Variation and their Interplay: An Integrated Divide-and-Conquer Method for Modeling Business Processes with Variation

DISSERTATIONES MATHEMATICAE UNIVERSITATIS TARTUENSIS
101

DISSERTATIONES MATHEMATICAE UNIVERSITATIS TARTUENSIS
101

FREDRIK MILANI

On Sub-Processes, Process Variation and their Interplay: An Integrated Divide-and-Conquer Method for Modeling Business Processes with Variation

UNIVERSITY OF TARTU
PRESS
1632

Institute of Computer Science, Faculty of Mathematics and Computer Science, University of Tartu, Estonia.

Dissertation has been accepted for the commencement of the degree of Doctor of Philosophy (Ph.D.) on June 15, 2015, by the Council of the Institute of Computer Science, University of Tartu.

Supervisors: Prof. PhD. Marlon Dumas
Institute of Computer Science, University of Tartu, Tartu, Estonia
Assoc. Prof. PhD. Raimundas Matulevičius
Institute of Computer Science, University of Tartu, Tartu, Estonia

Opponents: Prof. Dr Manfred Reichert
Faculty of Engineering and Computer Science, University of Ulm, Ulm, Germany
Prof. PhD. Jānis Grabis
Department of Management Information Technology, Riga Technical University, Riga, Latvia

The public defense will take place on 28th of August 2015, at 16.15 in Liivi 2-403.

This research was supported by European Social Fund's Doctoral Studies and Internationalisation Programme DoRa, which is carried out by Foundation Archimedes.

The publication of this dissertation was financed by the Estonian Doctoral School in Information and Communication Technology.

ISSN 1024-4212
ISBN 978-9949-32-859-8 (print)
ISBN 978-9949-32-860-4 (pdf)

University of Tartu Press
www.tyk.ee

To my family, friends and fellow Bahá'ís who labor ceaselessly
"to carry forward an ever-advancing civilisation"

ABSTRACT

Every organisation can be conceived as a system where value is created by means of business processes. In large organizations, it is common for business processes to be represented by means of process models, which are used for a range of purposes such as internal communication, training, process improvement and information systems development. Given their multifunctional character, process models need to be captured in a way that facilitates understanding and maintenance by a variety of stakeholders.

To achieve this goal, it is generally accepted that a complex business process should not be captured as a single large model but rather as a collection of smaller and simpler models following a divide-and-conquer approach. This advice applies particularly in the case of business processes that have multiple variants, such as an order-to-cash process that varies depending on the geographic region, customer type or product type.

For the purpose of modeling, a business process can be divided in two ways. On the one hand, a process may be split into subprocesses, such that their concatenation captures the entire process. On the other hand, a process may be split into variants, such that each variant captures end-to-end, a subset of the possible executions of the process. In other words, the process is the union of its variants.

The benefits of divide-and-conquer approaches to process modeling are widely acknowledged. Accordingly, a range of subprocess decompositon criteria and heuristics as well as process variant modeling approaches have been proposed. However, proposals in this field have largely been made in isolation of one another, leading to a lack of an integrated divide-and-conquer approach to modeling business processes with variants.

This thesis addresses this gap by proposing an integrated decomposition-driven method for modelling business processes with variants. The core idea of the method is to incrementally construct a decomposition of a business process and its variants into subprocesses. At each level of the decomposition and for each subprocess, we determine if this subprocess should be modelled in a consolidated manner (one subprocess model for all variants or for multiple variants) or in a fragmented manner (one subprocess model per variant). This decision is taken based on two parameters: (i) the business drivers for the existence of the variants; and (ii) the degree of difference in the way the variants produce their outcomes.

The method is validated via two real-life case studies: one concerning the consolidation of existing process models, and another dealing with green-field process discovery. In both cases, the method produced fewer models with respect to the baseline and reduced duplicity by up to 50%, without significant impact on the complexity of the resulting process models.

ACKNOWLEDGEMENTS

This is the end of four years of doctoral studies, resulting finally in a PhD thesis. These, have certainly been exciting years during which I have expanded my knowledge base, acquired new understandings, gained new insights and extended my experiences with fresh perspectives. This has only been possible through the supervision, accompaniment, support, encouragement and manifold discussions I have had with a variety of minds and souls.

First of all, I wish to place on record, my gratitude to my supervisors, Professor Marlon Dumas and Associate Professor Raimundas Matulevičius, for having shared their expertise and academic experiences, which have been instrumental for the completion of this PhD thesis. My papers would not have been accepted had it not been for Marlon's "magic touch". His revisions of sentences, turned scientific prose in academic poetry that conveyed exactly what needed to be communicated, in the exact manner required. Raimundas, on the other hand, patiently weeded out all the ambiguities that I had unintentionally planted. Needless to say, those valuable comments improved the quality of my papers considerably.

I wish to express my appreciation of Naved, who at the beginning was a colleague but is now, a close friend. Thank you for all the wonderful social and scientific conversations extending from where to have lunch to how to secure a world class LIMS.

I am also extremely thankful to Vesal for all the spontaneous discussions, his support, advice, company and for being available at a moment's notice whenever I needed his views or perspectives on something.

I also wish to take this opportunity to express my appreciation for all my former colleagues, in particular Birgitta and Lars. I deeply value all our conversations we have had and all your help with data and perspectives whenever I needed it. Your input has assisted me greatly and made its imprint on this thesis.

Also, I wish to extend a truly special thanks to my family. Words cannot express how grateful I am to my parents, my brother and his family and to Roya and Foad, for all of their support. Although I do not express it as often as I should, I am extremely grateful for all the different ways you have assisted and continue to support me.

Finally and most importantly, I would also like to express, with all my heart, mind and soul, my deepest gratitude, appreciation, and love to my beloved wife Lili and my children, Leona and Adrian. There is simply not enough space to convey how much you matter, how grateful I am for you bearing with me, for being there during the good times, the better times and the best of times.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ORIGINAL PUBLICATIONS

Milani, F; Dumas, M and Matulevičius, R. “Identifying and Classifying Variations in Business Processes. Enterprise, Business-Process and Information Systems Modeling, Lecture Notes in Business Information Processing Volume 113, Springer Berlin Heidelberg, 2012. 136–150.

Milani, F; Dumas, M and Matulevičius, R. “Decomposition driven consolidation of process models.” Advanced Information Systems Engineering, Lecture Notes in Computer Science Volume 7908, Springer Berlin Heidelberg, 2013. 193–207

La Rosa, M; Aalst, W van der; Dumas, M; and Milani, F. “Business process variability modeling: A survey.” Technical Report, accessible at http://bpmcenter.org/wp-content/uploads/reports/2013/BPM-13-16.pdf (2013), 51 pages.

Milani, F; Dumas, M; Matulevičius, R and Ahmed, N. “Modeling Families of Business Process Variants: A Decomposition Driven Method.” arXiv preprint arXiv:1311.1322 (2014) – submitted to Information Systems 2014, 27 pages.

Milani, F; Dumas, M; Matulevičius, R and Ahmed, N. “Criteria and Heuristics for Business Process Model Decomposition: Review and Comparative Evaluation.” Submitted to BISE Journal 2015, 21 pages.

I INTRODUCTION

Organizations, be it non-profit, governmental or private, operate in an increasingly competitive and changing landscape. In order to gain or maintain their competitive edge, they constantly seek to improve their efficiency. It is essential for organizations to constantly evaluate how they create value and identify opportunities for improvement, if they are to reach higher levels of efficiencies. A means to this end is by focusing on the value producing processes of an organization. Such approaches and methods fall within the *Business Process Management (BPM)* field.

BPM is "the art and science of overseeing how work is performed in an organization to ensure consistent outcomes and to take advantage of improvement opportunities" [54]. When embarking on a BPM journey, organizations need to ask what processes they currently have (*process identification*) [54]. A business process can be defined as a set of activities that together, produce a desired outcome or a business goal [190]. For instance, most organizations, such as an insurance company, have an order-to-cash process that covers the process from which an order is received until the ordered product or service is delivered.

Within BPM, the aim is to manage the business processes that add or produce value for the organization and its customers. As such, there is limited value to working with all business processes at the same time. Rather it is better to focus on a few processes, preferably those that are at the core to the organization and where improvements result in the greatest benefits for the organization and its customers. Therefore, the next step is to understand the selected business processes in more detail. For instance, the order-to-cash process of an insurance company will most likely include steps such as registration, issuing an insurance, creating an invoice and register premium payments. Each of these steps can be further detailed until such a level where further detailing of the steps do not add any further value. The work that is performed to graphically capture business processes as models, is called *process discovery* [54].

Once these business processes are describes as *business process models* depicting the current situation (also called as-is process models), they are analyzed (*process analysis*) and inefficiencies, waste and opportunities for improvements can be identified. For instance, the insurance company mentioned above, might notice that many customers contact the company to get insurance but do not complete the process (i.e. do not become customers). Further analysis might reveal that the customers provide the required data but it takes two days before their requests are approved. While waiting, the customers find other insurance companies that offer them insurance faster. More detailed analysis might further reveal that the delay is because one department receives the requests and another department processes them.

After the process analysis, the as-is process models are modified or redesigned to depict the desired state (also called the to-be process models). For instance, the insurance company might decide to have the same department process all insurance requests. These changes can then be implemented in the

business processes of the organization. Finally, the performance of the business process is monitored and further improvements or adjustments can be made as they are identified.

As such, process models play a vital part in BPM. In fact, the process models will be the main artifact for discovery, analysis and conceptual modifications or re-design of business processes. However, organizations will oftentimes find that their processes (such as order-to-cash in the example above) do not exist as singular entities but rather as a family of variants that need to be collectively managed [52, 168]. For instance, the insurance company might have a set of processes for managing claims (*claim-to-resolution*). During process discovery, when the process is modeled, one can observe that the insurance company typically performs the process for handling a claim differently depending on whether it concerns a personal, vehicle or property claim [150]. Each of these processes for handling a claim, is a variant of one generic claims handling process [72]. As such, there are variants of business processes that increase the complexity and needs to be managed in a structured manner.

1.1 Problem statement

When it comes to modeling a family of process variants, one extreme approach is to model each variant separately. These process models are very simple as they show the activities in a straight sequence and are therefore easy to understand. However, such a *fragmented-model* approach [52] or a "*multi-model approach*" [72] creates several problems. Commonly such models exhibit high level of redundancy as many models have fragments that are similar or even identical. In addition, models in such collections are loosely connected with each other and it is not always clear which variants are parts of one family. Another aspect to consider is maintenance and changes to the models. Given their large number and possible redundancy, it might prove to be both time consuming and error prone to keep the models up to date. Furthermore, it is possible that the models are optimized independently over time and therefore do not benefit from synergic effects [72]. However, it should be noted that in some cases, such a fragmented approach might be better (for example when a set of process models have few variants that are largely different and independent from each other) [72].

On the other hand, modeling multiple variants together in a *consolidated-model* approach [52] or "*single-model approach*" [72] has its disadvantages as well. This approach will result in one single process model that is relatively large, as it includes all variants. Such models rely on annotating variations with meta-data and thus facilitate management of the process variations from different perspectives. However, large process models with many variants baked into one model are very difficult to understand and to work with. Furthermore, these models capture and give the same importance to both frequently used process paths (variants) as to less frequently executed variants. This increases the com-

plexity of the models and further limits its usability in the daily work of business analysts. Such consolidated process models may therefore, prove difficult to analyze, evolve and maintain [116].

Striking a trade-off between modeling each process variant separately versus collectively in a consolidated manner is still an open research question.

Most of the annotation-based approaches, manage variability in process models based on the degree of similarity between the variants (measured by means of string-edit and graph-edit distance [45, 53]). In addition, annotation based approaches require that (*i*) the models of the separate variants are available; and (*ii*) that they have been modeled using the same notation, at the same level of granularity and using the same modeling conventions and vocabulary. These assumptions might not hold in many practical scenarios where models of each variant might not be available to start with, and even if they were available, they would typically have been modeled by different teams and using different conventions. Therefore, there is a need of a systematic approach to manage process variability in consolidated business process models.

In this context, the main research questions is "How can a family of process variants be modeled when consolidating or discovering business process models?"

1.2 Scope and Limitations

The standard business process life cycle consists of four main stages and variability needs to be managed at each of these stages. The first stage is *design* of the business process where families of business processes are designed [71]. At this level, the process models capture all variants of a family of business processes. Decisions at this stage of the life cycle have significant impact on the business processes. The second stage is often termed *configuration* or *customization* phase [71]. At this stage, a customized process model is created, describing one single variant of the family of process model. The third stage is when a customized process model is being *deployed* or *instantiated* in its run-time environment [71]. Execution alternatives at this level are not pre-determined (as opposed to design time) but dependent upon requirements at run-time [30]. The fourth stage concern monitoring and optimizing the business processes i.e. managing feedback and improvement of the business processes [71]. At this stage, the variability of business processes is more related to measurements.

The scope of this thesis is delimited to *design-time variability* i.e. when configurable or customizable process models are created. There are approaches, methods and principles for or related to managing process models that deal with variability. These approaches, methods and principles such as process model queries, refactoring, abstraction and cognitive aspects of process modeling, while offering valuable insights to the body of knowledge on managing process models, are not designed for managing variability but consider variability, as it

is an integral part of business processes. They are therefore excluded from the scope of the thesis.

1.3 Contribution

The first contribution of this thesis stems from a review of the state of the art on approaches to manage variability in process models. In this phase, it was observed that many approaches have been proposed for manage variability in business processes. However, there is no categorization of the approaches that will guide the users in choosing the most appropriate approach or to easily get an overview of how the approaches are positioned against each other. As such, the first contribution of this thesis is:

- A systematic review (state of the art) of current approaches to manage variability in process models and a framework to classify such approaches (chapter 2).

The systematic review highlighted the need to manage variability as a means to reduce complexity. Furthermore, it was clear that variations are mostly managed based on their degree of similarity and as such, emphasis is not given to the underlying business reason for the occurrence of variability.

Variability increases the complexity of business processes, in particular when the number of processes and process variants grow. This complexity is augmented when trying to represent them in process models. A common approach to manage such complexity is to decompose process models into more manageable parts. Although a variety of approaches exist on how to decompose a process, it is not clear on what basis such vertical and horizontal decomposition should be made. It is necessary to identify decomposition methods that can be applied on a set of process models in order to manage the complexity arising from variations. As such, the second contribution of this paper is:

- A review, classification and analysis of decomposition approaches and introduction of a variant-driven vertical decomposition of process model (chapter 3).

When decomposition of process models are based on similarity-based parameters, as the systematic review revealed, the root causes of variations are not considered and there is a higher risk of creating a distance between the models and the business reality they aim at representing. The underlying business reasons for variations, as complimentary to similarity-based parameters, needs to be considered when decomposing process models. Failing to do so will increase the risk of alienating the model from the reality it aims to represent. In order to incorporate business reasons when deciding how to decompose processes, we need to identify and categorize these business reasons. The third contribution of this thesis is therefore:

- A definition of variable variants, business drivers and an orthogonal classification of root causes of variability in business processes (chapter 4).

However, most of the methods for managing variability when consolidating or discovering process models are only applicable if and only if certain prerequisites are met. In addition to not considering the business drivers, inconsistency of modeling convention and differences in granularity of detail, pose a challenge when managing process variability. In order to manage variability that is "closer" to the business reality by considering their business drivers when certain required inputs are not available, there is a need for a method to model families of process variants. As such, the forth and the main contribution of this thesis is:

- A decomposition driven method for managing family of process variants (chapter 5).

The core idea is to incrementally construct a decomposition of the family of process variants into sub-processes. At each level of the process model decomposition and for each sub-process, it is determined if a sub-process should be modeled in a consolidated manner (one sub-process model for all variants or for multiple variants) or in a fragmented manner (one sub-process model per variant). This decision is taken based on two parameters: (*i*) the business drivers for the existence of a variation in the business process; and (*ii*) the degree of difference in the way the variants produce their outcomes (syntactic drivers).

The applicability of the method was verified with two in-depth case studies from different industrial settings with the main research question of "*how can a family of process variants be modeled*? The case study method was chosen, as it is a suitable method of evaluation used within software and system engineering domain. In addition, case study method is particularly useful when the case requires frequent, intensive and prolonged interaction with domain experts. The modeled family of process variants should aim at minimizing the total size and duplicity while not causing overly complex process models. Furthermore, the process models should be aligned with the underlying business processes. In light of this, the final contribution of this thesis is:

- Validation of the method on two distinctly different case studies where the results verify the purpose and objective of the method (chapter 6).

The State of the Art (chapter 2) is an adapted extension of La Rosa, Marcello; Aalst Wil van der; Dumas, Marlon; and Milani, Fredrik. "Business process variability modeling: A survey." Technical Report, BPMCenter.org, (2013). In this paper, I conducted the systematic literature survey and participated in classifying and examining various approaches.

The section of foundations of process decomposition (chapter 3) is an adapted extension of Milani, Fredrik; Dumas, Marlon and Matulevičius, Raimundas; Ahmed, Naved. "Criteria and Heuristics for Business Process Model Decomposition: Review and Comparative Evaluation." Submitted to BISE March 2015. I am the main author of this paper.

The section of foundations of process variants (chapter 4) is an adapted extension of Milani, F; Dumas, M and Matulevičius, R. "Identifying and Classifying Variations in Business Processes." Enterprise, Business-Process and Information Systems Modeling, Lecture Notes in Business Information Processing Volume 113, Springer Berlin Heidelberg, 2012. 136–150. I am the main author of this paper.

The method proposed in the thesis and the case studies (chapter 5 and 6) are based on extended versions of the following two papers:

Milani, F; Dumas, M and Matulevičius, R. "Decomposition driven consolidation of process models." Advanced Information Systems Engineering, Lecture Notes in Computer Science Volume 7908, Springer Berlin Heidelberg, 2013. 193–207. I am the main author of this paper.

Milani, Fredrik; Dumas, Marlon and Matulevičius, Raimundas; Ahmed, Naved. "Modeling Families of Business Process Variants: A Decomposition Driven Method." arXiv preprint arXiv:1311.1322 (2014) – submitted to Information Systems 2014. I am the main author of this paper.

1.4 Structure of the thesis

This thesis is structured as follows. In chapter 2, a comprehensive literature review of approaches to manage process model variations is presented to position the contribution of this thesis against the state of the art. The approaches are organized in accordance with a proposed classification framework. Furthermore, the advantages and disadvantages of each category of approaches are discussed.

Chapter 3 presents the foundations of process decomposition where the ideas on how to decompose business processes (decomposition heuristics) are examined and categorized.

Chapter 4 presents the foundations of process variation. Here, the concept of a "viable variants" of a business process is defined, followed by elaboration of different drivers for variations in business processes.

Chapter 5 operationalizes the foundations presented in chapter 3 and 4 into a decomposition driven method for managing families of process variants.

Chapter 6 presents the application of the decomposition driven method to model families of process variants on two case studies, one for process model consolidation and one for process model discovery. In addition, the findings from the case study are presented and analyzed and finally the threats to validity are discussed.

Chapter 7 concludes this thesis by summarizing the work presented, discussing the contribution of the thesis and pointing out the direction of future work and possible alternatives for extensions.

2 STATE OF THE ART

The purpose of this chapter is to (1) review current relevant approaches to manage design-time variability in business processes and process models, (2) to propose a framework for classifying these approaches and finally (3) position the research of this thesis in relation to the reviewed approaches.

2.1 Categorization of Approaches to Manage Business Variability

2.1.1 Search process

The literature search process was conducted based on the principles of systematic literature review according to Kitchenham [88]. The process started by submitting queries to a well-known research literature database (Google Scholar) covering the main keywords associated with the scope of the survey. The search, conducted in March 2013, included the terms

- "Customization" associated with "variation" and "configuration".
- "Business process" with "customization", "variation", "configuration", "customizability", "variability", "configurability", "customizable" and "configurable".
- "Flexibility", "flexible" and "flexibility" associated with customizability.
- "Business process variant", "configurable reference model", "reference model adaptability", "reference model flexibility" and "configurable EPC"
- "Software Product Line Engineering" (SPLE), software product line", "feature model" and "UML activity diagram"
- "Workflow" combined with above listed keywords.
- "Business Process" and "Process Model" in combination with "standardization", "method", "consolidation", and "framework".

For each query, the first 50 hits in Google Scholar where considered for the first filtering based on the title in order to eliminate papers that were clearly not related to the topic. Following this, a second filtering was performed to remove duplicates, papers with no citations, as well as short papers (less than 5 pages) as they would not contain enough information for an evaluation. During the next step of filtering, the abstracts of the papers were inspected. Papers that did not cover design-time variability (such as run-time variability and exception handling) were excluded. The filtering process resulted in a total of 95 relevant papers that cover approaches presented in this chapter.'

2.1.2 Excluded Approaches

There are many approaches that deal, either directly or indirectly with business processes and process models. As variability is an integral part of business processes, these approaches need to consider variability. As such, these approaches are not designed or created for managing variability but manage them with a different objective in mind. This literature review, therefore excludes approaches of such kind. Approaches that have been excluded from this literature are listed below.

Querying in repositories of process models focus on methods for identifying process models or fragments of process models. The purposes of querying vary. For instance, it could be for the purpose of retrieving models that have specific attributes or for analyzing if process models comply to certain standards [48]. A set of related approaches are *similarity searches* in repositories of process models. However, while querying returns exact matches, similarity search will return approximate matches [48]. Both search and return a set of process models from an input of process model. For whatever purpose, these approaches focus on finding a fragment of a process model (identical or similar) and are therefore, excluded.

Process Model Refactoring is an application of refactoring from software engineering domain where code or databases are re-structured without changing their behavior. In this light, approaches that apply the same principle on the domain of process models have been developed [48]. Refactoring does not manage variability explicitly as these approaches focus on identifying and restructuring fragments of a process model in order to improve for example maintainability. For this reason, refactoring approaches have been excluded.

Approaches that manage abstraction of process models, focus on representing business processes at different levels of granularity. This is mainly motivated by different interests in level of detail that various stakeholders have. For example, top level management are perhaps not as interested in detailed process models as those who work with the processes on a daily basis. Abstraction methods aim at managing different levels of abstractions of business processes by managing hierarchical model structuring that permits organizing process details at different levels of detail [128, 130]. Although abstraction approaches include variability, they do not explicitly manage variability and are therefore excluded from this review.

Model synchronization refers to approaches that seek to consolidate different versions of process models based on identifying and resolving the differences between them [63, 64]. These approaches are excluded as they take two different versions of the same business process as input and not two design-time variants of a business process.

A number of approaches such as [2, 80] deal with managing evolving workflows. These approaches are focused on how to manage cases where the business process is evolving rather than variability in the business processes. Due to this reason, they are also excluded from this review.

A number of business related approaches such as six sigma [25] and TQM (total quality management) [79] are also out of the scope of this review. These methods do not seek to manage variability but rather focuses on incrementally improving processes in order to make them increasingly efficient.

Finally, papers focused on improving understandability of process models such as modularization [197] or work based on cognitive understanding of process models [174, 175] are also excluded. These papers deal with understandability of given process models and not managing variability to increase understandability.

2.1.3 Necessity for classification

The systematic literature review described above resulted in a list of various approaches where variability in process models is managed. However, the list of approaches is long and does not lend itself to be easily understood or analyzed for the purpose for examining the relations between different approaches or for positioning the method proposed in this thesis. To remedy this, a framework for classification of approaches managing variability is proposed in this thesis. The parameters used for the classification are, (1) if they propose consolidating or fragmenting the process models and (2) if they serve the purpose of re-organize the process models or if they aim at effecting a change in the business processes. The classification was born from analyzing the many approaches to manage variability.

When reviewing the approaches, it became clear that a set of approaches adopt the philosophy of separating variants by modeling as separate process models while others, started with the notion of modeling all variants in one or few process models. Following this initial standpoint, the approaches propose managing variability by either moving to fewer process models or by dividing larger process models into several process models. As such, a set of approaches move from fragmented to consolidated process models while other approaches move in the opposite direction. This observation caused the definition of the first axis of the classification namely – consolidating or fragmenting process models.

Furthermore, commonalities were distinguished among the approaches in regards to their purpose or objective. Some approaches focused on affecting a change in the actual business processes by modifying, improving or creating new business processes. Although this objective is achieved by means of process models, the main artifact is the actual business process. On the other hand, a set of approaches focus only on managing the process models that were available for the purpose of improving understandability and maintainability of the process models. This observation caused the identification of the second axis of the classification – re-organize process models or change business processes.

2.1.4 Fragmenting versus Consolidating

The first classification parameter is related to how the approaches manage variants of a process. At one extreme, each process variants can be seen as a distinct process and therefore, they are modeled and managed separately. As mentioned in the introduction, such collection of process models creates redundancy, as several variants of the same process, will have the same activities or even sets of activities. Furthermore, when changes are made to a process model, it can create inconsistencies if the changes are not applied in all other process models. Ensuring consistency requires effort and as such, the maintenance of fragmented-model approach is both time-consuming and error-prone.

On the other hand, multiple variants can be managed collectively in a consolidated manner where they are all modeled as one large process model. In such cases, the variants are all captured in one process model. Variants are represented through gateways such as XOR and OR splits. Such an approach will usually result in large and complex process models, which will cause understandability and maintainability issues. Furthermore, one should bear in mind that, with all variants are integrated in one model, it will become increasingly difficult to distinguish the main flow from exceptions [72].

All reviewed approaches, assume a starting point on this continuum and propose a set of changes that will either make the collection of process models more "fragmented" or more "consolidated". Variability is therefore managed by a set of operations that either reduces (restricts) or creates more (extends) the number of models. If the approach proposes a reduction, i.e. starts from a fragmented and moves towards the consolidated end, it is termed "*standardization*". Standardization entail that several process model variants are merged into one process model. However, if the approach takes the opposite stand, i.e. manages variability by extracting a variant from a consolidated model and thereby extending the number of models, it is termed "*customization*". Customization entail that a consolidated model is customized to extract or create a specific process model variant.

2.1.5 Business Process versus Process Model

The second classification parameter, used to classify approaches, is related to which reality it proposes to change, the business processes or the process models. Some approaches are designed and have the purpose to create or modify business processes whereas other approaches focus on improving existing process models for improvement purposes.

For instance, the previously mentioned insurance company might have two different variants for managing its "issue-to-resolve" process, one for corporate clients and one for private clients. Let us assume that they have identified benefits in terms of lowered costs if they treat these variants in the same way. They, therefore, wish to replace these two variants with one process. The approach employed, will aim at improving their business process, i.e. the end result will be an alternation or change in the actual business processes of the insurance

company. These approaches, therefore, aim at creating one business process to replace two or more existing business processes, often for the purpose of achieve better efficiency and as such, cause a change in the business processes.

On the other end of this dimension, are approaches that re-arrange and re-structure existing process models for the purpose of making them more comprehensible and manageable. The operations of these approaches are limited to process model. These approaches seek to improve the representation of business processes by improving the quality of process models. As such, they cause an alteration or change in the process models but not the business processes these models represent.

2.1.6 Classification of Approaches

The two classification parameters defined under the previous headings, when juxtaposed with each other, creates four quadrants (cf. Fig. 1). All the reviewed approaches in this thesis are classified in one out of four quadrants.

The first quadrant, "*business process standardization*" encompasses approaches that will result in changes to the business processes by moving from multiple (fragmented) processes towards fewer (consolidation) through re-design, merger or replacement.

The second quadrant, "*business process customization*" covers approaches that will result in changes to the business process (update or creation) by extracting new variants or modifying variants from consolidated processes.

Conversely, the third quadrant, "*process model standardization*", refers to approaches that will cause a change the process models by reducing the number of variants (moving from the fragmented towards the consolidated end).

Finally, "*process model customization*" embraces approaches that will only affect the process models by extracting existing variants from consolidated process models.

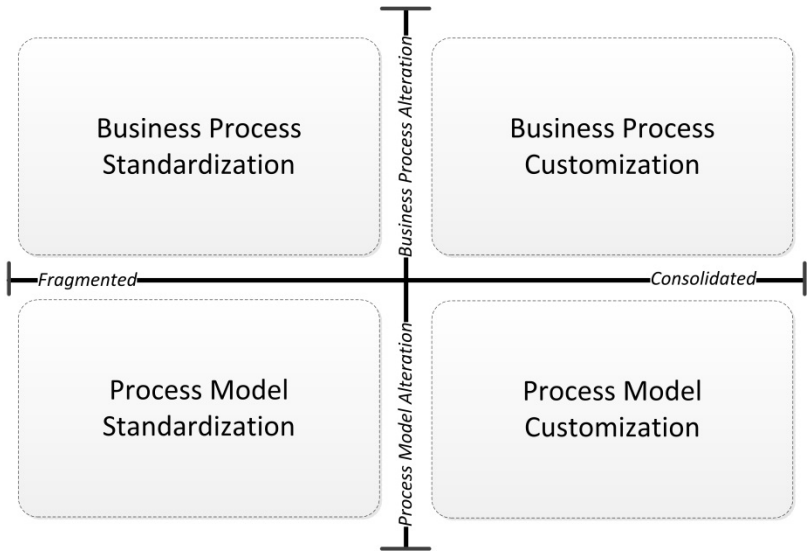


Figure 1: Classification Framework for Approaches Managing Variability.

It should be noted that an approach, classified in one of the quadrant, could be used for another purpose. For instance, a configurable process model is categorized as "*business process customization*". However, it is possible to create a configurable process model from existing variants and use it for "*process model customization*". This would mean that it is used to manage process models rather than creating new business processes. However, the approaches are developed as a response to an identified problem or challenge. Therefore, the classification has been made based on the primary use of the approach.

2.2 Business Process Standardization

Business process standardization, in this context, refers to reducing the number of variants of a certain business process. In order to reduce the number of variants, it is many times necessary to understand the existing variants by modeling them. The modeling of business processes is commonly referred to as "process discovery" [54]. It is worth noting that all approaches in the other quadrants, assume that there are process models to being with. This is not necessarily true with this quadrant. As such, a brief overview of process discovery methods follows.

2.2.1 Process Discovery

Methods for process model discovery can broadly be classified into automated or manual. Automated methods exploit existing data to generate a process model. In this category, one sub-category of approaches is concerned with the

discovery of process models from execution logs, also called "process mining" [1] or "workflow mining" [9, 11]. Some of these approaches use trace clustering to uncover potential variants of a process, arguing that variants would manifest themselves as clusters of similar traces in the logs.

Another sub-category of approaches for automated process discovery is based on textual documentation. For instance, Ghose *et al* [65] propose a framework for Rapid Business Process Discovery (R-BPD). Their framework is based on querying text artifacts, such as corporate documentation, to create initial process models that are subsequently edited by domain experts.

Non-automated process model discovery methods are concerned with collecting, organizing and analyzing data from various stakeholders as source of information for producing process models. The method suggested by Sharp and McDermott [167] is an exemplar of a method in this field. Another similar method is presented in [121]. These methods employ a set of guidelines that are used when eliciting the business process together with domain experts, most commonly in workshop settings. These approaches recommend modeling variants separately in a "fragmented" manner.

The method for process identification defined in Dumas *et al* [54] is also an example of non-automated method. This method begins with the identification of cases (variants) and functions that should be included in the process architecture. Next, a case/function matrix is created and by applying a set of 8 guidelines, processes are identified from this matrix. Two steps in this method explicitly deal with variations. In the first of these steps, variants of a business process (called cases) are listed. Later, in a second relevant step, if a process model for one case is found to be syntactically very different from the model of another variant, the two variants are explicitly separated.

2.2.2 Standardization

Many organizations have several processes that produce similar outputs. Consider, for example, a multi-national corporation that has a procurement process for each of the markets it is operating in. Managing and maintaining several variants of the same process, is considered as costly [176] that can be expressed indirectly (customer dissatisfaction, inefficiencies, ineffectiveness) or directly (IT development and support) [176] in the cost structure of a company. The foundational base of approaches seeking to standardize is that "one process version is better than many" [75]. By standardizing business processes, several business processes (variants) are reduced and replaced with one single business process [142, 176].

2.2.3 Selecting Processes for Standardization

One of the challenges within the context of business process standardization is selecting processes to standardize. Hall & Johnson [70] assert that all processes

are not optimal for standardization. They introduce a matrix that helps managers in the elicitation of candidate processes for change and those best left alone. In their matrix, they consider process environment being either of low variability or high variability. They also consider if variants have a positive or negative value for the customer. If a process has high variability and the customer positively values variations in output, the process is seen as an "*artistic processes*". Such processes, (e.g. customer service on first class flights), should not be standardized. Conversely, processes with low variability were variation in output is perceived by the customer as negative, are termed "*mass processes*". These processes, such as consumer financial services, on the other hand should be highly standardized.

Schafermeyer *et al* [160–162] present a conceptual model depicting the relationship between process complexity, standardization effort and process standardization. They show that there is a negative relation between standardization effort and process standardization, meaning that by simply putting more effort to standardize, does not necessarily give more standardized processes. They also show, in line with Hall & Johnson [70] that highly complex processes are not necessarily possible to standardize. Schafermeyer *et al* [161] adapt and introduce new measures for assessing process complexity, standardization effort and process standardization. In total, there are 19 measurements. The measurements are formed as statements that BPM experts will agree or disagree with for a given business process. Depending on the level of agreement, a business process can be measured for complexity, standardization effort and level of standardization. For example, one of the statements regarding process standardization is, "during the execution of the business process we follow a well-regulated process cycle".

Rosenkranz *et al* [154] also question the value of standardizing complex processes. In their case studies, they found that some processes have fragments that are routine while others are complex. In line with other authors [70, 161], they state that it is not desirable to standardize complex parts of the business process. However, those sections that are well structured can be standardized in order to release time, allowing the actors to focus more on the creative parts of the business process. They conclude that an analyst need to understand if a business process can be standardized as a whole or if some sub-processes (fragments) are more suitable for standardization. Furthermore, they also conclude that the analyst must consider the purpose of the standardization initiative in order to assess what aspects of a process should be standardized.

Other researchers have proposes a variety of factors and contexts that determine if and when variants of a business process can and should be standardized. Ang and Massingham [13] suggest considering the national culture that multinational corporations operate within in order to determine if it is appropriate or inappropriate to standardize. Girod and Bellin [66] investigated the context of emerging-market multinationals and their challenge of standardizing their operations (global integration) or keeping local variations (local responsiveness). They propose considering how the companies' organizational

capabilities have evolved when considering when to globally integrate and when to have local variations.

2.2.4 Merging Process Models

Hammer and Champy [74, 76] together with Davenport [37, 38] introduced and developed the concept of Business Process Reengineering (BPR). The main idea of business process reengineering is to consider the whole process and radically change to create dramatic improvements in efficiency, cost reduction, time to delivery and customer satisfaction. BPR grew in popularity in the beginning of 1990-ties which caused a myriad of consultancy firms to offer BPR services. BPR does not offer a method on how to identify and radically change business processes but rather guidelines and general principles. As such, this has lead to a proliferation of methods and techniques [85]. The most common denominator [85] of all BPR approaches are (1) Envision – establish management commitment and vision, (2) Initiate – informing stakeholders, (3) Diagnose – document existing processes (process discovery as described previously), (4) Redesign – define and analyze new processes (such as reducing the number of variants), (5) Reconstruct – reorganize and implementation and finally (6) Evaluate – evaluation of process performance. Most BPR approaches include most of these steps but vary in the techniques proposed for conducting each step. However, these approaches all are classified as "business process standardization" because they seek to improve existing business processes (variants) by replacing them with one standard business process and as such, they work with improving business processes by reducing their number.

Ludwig *et al* [104] proposes dealing with variability in the context of business process standardization in four steps. The first step, "scoping", refers to determining which business processes to include in the project. The second step, "variant identification" concerns not only identifying the variants but also understanding why they exist. The third step is "variant adjudication" refers to the evaluation and determination if a variant is to be included in a standardization effort or not. Finally, in the fourth step, "change implementation", the standardization is implemented. Ludwig *et al* [104] do not provide much detail about each step. They implement a Work Practice Design (WPD) method to elicit variants and thus focusing more on their second step (variant identification). WPD, in similarity with other user-centered based approaches, use a range of data collection methods (such as various forms of interviews and observational studies) for the purpose of uncovering how people work with resources in order to achieve the business goals.

Ungan [181] propose a framework for standardization based on process documentation. He targets mainly processes that are largely dependent on the tacit knowledge of the employers. In his framework, the first step is to identify the processes to be standardized. He recognized that not all processes can be standardized and state that those processes that share identical inputs and out-

puts can be standardized. The next step is to identify those individuals who master the process i.e. who have knowledge about how to perform the tasks of the process. Then the processes to be standardized, are detailed in terms of purpose, boundaries, list of customers, suppliers and so on. When this is completed, the next step is to acquire knowledge for each step. During this step, the tacit knowledge of the employees are externalized and documented with the aid of knowledge management methods. At this stage, the standardized process is worked out. Following this, the knowledge about each step is documented, clarified, verified and agreed upon. Effort is made to ensure consistency of the terms used and that each term is understood the same way by all participants. As a final step, all these process documents for each step are combined to produce single standardized process documentation.

Manrodt & Vitasek [108] examined the logistic process of a global company and introduced a framework for standardizing such processes. In their framework, the two initial steps are "articulate strategy" and "process view of logistics". As a first step, it is necessary to have an articulated strategy to standardize, accepted by both management and employees. Given this strategy, it is necessary to adopt a process view of logistics, meaning that involved persons should start thinking in terms of processes and move away from thinking in functional silos. With these two steps in place, the third step "identify segments, processes and process attributes" concerns identifying what processes are to be standardized. Following this, the "customer impact" of the identified processes is determined. This will enable finding, for example, areas that can give the best return for the efforts made. The next step, "select key segments for improvement" refers to actual selection of those processes or segments of processes that are to be standardized and implemented. The final step, "identify and train global segment owner" is about deployment of the standardized processes by training those in charge at each unit of the company.

Tregear [176] propose a framework for standardizing processes that considers local variations. He acknowledges that although organizations wish to implement standardized processes, there are circumstances (for example geographical and cultural) that make a strong case for having local variations. In order to resolve this dilemma, Tregear [176] introduce a three-level standardization trajectory from the current state, a target state and finally a global standard. The current state depicts the current situation with variations and when these processes have been improved to reach the global step, they cease to exist, as they are identical with the global standard. Target state represents improvements of the current state but falling short of a global standard. Although the target state should ideally reach the global standard, it might not reach such a state if local variations have been accepted as valid. In such cases, the current state and the target state would be coincidental.

2.2.5 Process Harmonization

Business process standardization seeks to reduce variants by replacing them with one business process. Business process harmonization, on the other hand, recognizes that it might be more optimal if the variants are reduced to fewer business processes rather than to one business process [47, 110, 142].

Romero *et al* [110] propose a harmonization framework that allows for finding a better level of harmonization. The level of harmonization is influenced by the variability of the processes and the factors causing variability (the reason for the existence of variants). The variability of a certain process is measured using a set of metrics. The first metric is to determine the number of variants. Two processes are considered to be variants if they differ in at least one element such as activity or resource. The second measure is, on average, how often a certain element occurs in a variant. The third way of measuring the level of harmonization is by computing how often each activity is connected or associated with the same element. These measures are applied on activities, control flows, applications and resources and they aim at assessing the level of similarity between variants. It is also necessary to identify (through interactions with domain experts) what contextual factors drive these variants. By this is meant, identifying the causes of the variants. When the factors of variability and the metrics assessing the variability are identified, these are used as basis for analysis to determine the level of harmonization. This analysis will allow practitioners to identify what factors affect the level of harmonization and what processes have potential for improvement by being harmonized.

2.3 Business Process Customization

The second quadrant is "business process customization" and refers to approaches that assist in creating or changing business processes by using consolidated business processes. These approaches share the commonality that they start with a consolidated process model, from which new variants are extracted. In other words, they manage variability by extending the number of process models. The consolidated process models of these approaches do not necessarily include exiting process models but rather "reference models" or models depicting all possible variants. They act as an aid for the business analyst to design new business process (variants) by working with creation of process models.

2.3.1 Configurable Reference Models

The common denominator for configurable reference models is that they represent business processes that can be applicable in several different cases. The starting point is a model that covers all the variants. A specific variant or process for a specific scenario can then be created by fading out those elements that are not relevant [20]. For example, process based software packages may pro-

vide a reference model that can be used as a template for designing processes for a specific implementation of the product. The assumption is therefore, that reference models contain information that can be used for multiple application scenarios (such as the SAP R3 models). The work of customizing a reference model, involves fading out elements that are not relevant. Having done this, semantic correctness can be ensured by applying algorithms such as the one developed by Delfman *et al* [41]. This algorithm remove faded elements and ensure correct connections for the remaining elements. However, it is very difficult to anticipate all possible configuration requirements when developing a reference model and therefore there is a need for adaptations that are not covered in the reference model. Addressing this challenge, Becker *et al* [21] propose an approach with generic adaptations in addition to configurative adaptations. These are aggregation (adding new model fragments), instantiation (inserting values into the provided placeholders), specialization (adding, removing or changing elements) and conclusion by analogy (possible reuse of model structures as seen by users).

In Aalst *et al* [4] the need for being able to configure reference models is described. For instance, SAP reference model, the modeling notation of Event Process Chain (EPC) is used but the notation language cannot capture configuration aspects satisfactory. They show that reference models have an inherent problem of keeping the configurator in the dark as to what options are available, even possible or the relation between two or more choices. The reference model includes all functions, and as such there are probably several functions that are mutually exclusive but not possible to represent with EPC. This limitation of EPC is addressed with the proposition [4, 153] of extending EPC. The extension of EPC is called Configurable EPC (C-EPC). With the C-EPC, a reference model captures all the variations and serves as inspiration and support for the work of configuring actionable processes. Each gateway, called configurable nodes, is assigned a set of configurable alternatives (XOR, AND or OR). Each alternative refers to at least one process variant. Within a reference model captured in C-EPC, there are constraints called configurable requirements that are captured by tags that can be restrictive or a guideline. The configuration is done by assigning each node, with one alternative and thus restricting the behavior of the process. The configurator has both the restrictive and guiding tags as aid in this work. Individualization, that is extraction of a process from the reference model (refereed to as customization in this thesis), is made and all alternatives that are no longer valid are removed. The analyst need not worry about correcting the individualized process model, as mutually exclusive paths are already restricted.

C-EPC as described above, extends EPC with the concepts of configurable nodes and configurable functions. In traditional process modeling, configurable nodes correspond to the control-flow and activity perspectives. However, a process model also includes data perspective that depicts the data objects and the resource perspective that captures the organizational perspective of a process. These perspectives are not covered in C-EPC and in order to address this limi-

tation, La Rosa *et al* [148, 149] introduce C-iEPC. In C-iEPC, objects are shown to the right of the function and can capture information or physical object as input and output of the function. To the left of the function, the resources are captured and they can be manual (performed by a human), automated (performed by non-humans) or semi-automated (performed by both). The data and resource artifacts can also have connectors. For example, a function might need a certain input but can produce either one or another output. These alternatives are for instance depicted with and XOR node in the output data artifacts. C-iEPC therefore extends the C-EPC by applying configurable nodes, in addition to functions, to both roles and objects. They also provide an individualization algorithm that ensures the syntactic correctness of the individualized C-iEPC:s.

Another approach that can be said to be subsumed by C-EPC is presented by Korherr & List [91], which extend UML activity diagrams to capture variability. In their approach, variability can be defined at the level of an atomic task, a group of tasks (called an activity partition) or a gateway (a fork node in UML) in an activity diagram. A task or activity partition can be defined as being mandatory (the task or partition must be retained during customization) or optional (the task or partition can be excluded during customization). A gateway can be defined as being an alternative 0..1 choice (one or multiple outgoing flows can be selected during customization) or alternative 1..* choice (only one outgoing flow is selected). This method allows stating that the selection of an element (task, partition or flow) during configuration requires the selection of another element elsewhere (called a "requires" dependency), or that the selection of an element excludes the selection of another one elsewhere in the model (called an "excludes" dependency).

An approach similar to C-EPC is presented by Moon *et al* [117] by the name of Business Process Family Model (BFPM). They propose a two-level approach to capture customizable business processes. The first level manages activities. At this level, an activity (can also be applied to a sub-process) can be defined as common (mandatory) or optional (can be omitted during customization). The second level manages gateways. A customizable gateway can either be boolean (exactly one variant should be selected), selection (at least one variant should be selected) or flow (defines how different variants are to be executed sequentially, in parallel or as decision). It is also possible to define dependencies between variants. If for example, a variant is chosen for a given variation point, it can restrict the choice of variants for another variation point. Dependencies can be between variation points, between variants or between variation points and variants. Besides these features that are also supported by C-EPC, a gateway can be classified as either open or closed. A closed type restricts the choice of variants to those already identified whereas an open type allows the introduction of new variants during customization.

Nguyen *et al* [120] present an approach along similar lines but in the context of BPMN. In the approach of Nguyen *et al*, both activities and data objects can be made customizable, as well as message flows that connects activities in different pools of a BPMN model.

2.3.2 Configurable Workflows

Gottschalk *et al* [67] introduce hiding and blocking of activities as another approach to manage customization of configurable process models. C-EPC only provides a partial solution as it is only applicable to EPC notation language but Gottschalk *et al* [67] look at the same issue from a notation language independent perspective. Their approach is based on inheritance i.e. restricting a configurable model rather than adding functions to a process model. This is achieved by two restriction techniques, hiding and blocking. When a function is blocked, the execution of that activity is disabled but when a function is hidden, it disables that function while the execution of the path is still possible.

2.3.3 Application Based Domain Modeling (ADOM)

Reinhartz-Berger *et al* [23] argue that organizations with many different units need to have some degree of standardization in order to function as a single business unit. At the same time, each unit has its specific needs and conditions and therefore, their processes cannot be the same as for another unit with different needs. In order to address these conflicting objectives, they propose an "*organizational reference model*" that is generic and can provide business logic to be applied across all units at the same time as it can be customized for local needs. For this purpose, the authors present Application-based Domain Modeling (ADOM) as a platform for creating organizational reference models. ADOM is a three-level architecture solution to customize process models. The first level (language), hosts the meta-models that can be used to describe business process models, e.g. in EPCs. The second level (domain), hosts the customizable "reference" process models, which serve as templates for a particular domain, e.g. logistics. Finally, the last level (application), hosts the customized process models for specific companies, which can be directly derived from the reference process models at the domain level.

Variability is expressed with annotations. The attribute of type <min,max> is used to denote the elements. The min and max states the number of times the element can occur in individualization. If an element is tagged with <0,1>, the element is optional and can therefore be excluded. If, however, an element is tagged with <1,n>, it is mandatory and can be instantiated n number of times. Likewise, if a tag states <1,1>, then the element is mandatory and can occur only once. The default tag is <0,n>, which represents no constraints. This means that common elements (commonalities) are mandatory and allowed variants (optional) can be instantiated n number of times. In the customized models, it is possible to trace back to the reference model as they have reference model classifier (annotations). It is possible to add application-specific elements in the customized process models to meet specific needs. However, these additions are not reflected in the reference model. This approach has been applied with UML models [141], EPC [140] and BPMN [23] notation languages.

2.4 Process Model Standardization

Process Model Standardization encapsulates approaches that seek to improve a collection of process models (as opposed to business processes) by operations that reduce the number of process models. As such, these approaches do not affect the actual business processes but aim at improving the process models representing a set of business processes. Their objective is to reduce duplicity, improve comprehensibility and reduce maintenance efforts. These approaches manage variability by reducing the number of process models.

2.4.1 Similarity Based Approaches to Manage Variations

Uba *et al* [180] seek to address the problem of many duplicates in large repositories of process models. As organizations become more mature from a business process management perspective, or if for example mergers take place, the repositories of process models grow. It is common that the number duplicates grow in these repositories as new process models are extended or created. These duplicates, referred to as clones by the authors, impair maintainability. The authors present an indexing structure that supports fast detection of clones in large repositories of process models for the purpose of refactoring into separate sub-processes. Their method is based on Refined Process Structure Tree (RPST) and code-based graph indexing. The process models are taken as input and are representing as a tree of hierarchy of single entry single exit (SESE) fragments. Then the process models are indexed and duplicate SESE fragments (clones) are identified. The identified clones can be refactored into sub-processes and thereby increasing the maintainability of the repository of process models by reducing the duplicates. Based on this, Ekanayake *et al* [58] identify approximate clones as opposed to an exact clone. An approximate clone is a pair of similar fragments. Their method is based on the same assumptions as clone detection (SESE fragments) but measures the distance between two fragments. If this value is below a certain threshold, they are considered as candidates for approximate clones. However, such a results are of limited use if it fails to identify opportunities for standardizing or refactoring with a limited amount of change operations. Therefore, for refining the identification of approximate clones, a given medoid is used, as reference (for the cluster of fragments), to identify approximate clones.

Jung and Bae [83] also propose a similarity based method for variability management. Theirs is a two-step approach. The first step is clustering processes based on their similarity. This clustering of models is based on activity similarity measures. If a set of process models shares common activities, they are considered to belong to similar domains and are therefore clustered together. Then, as the next step, the models of a domain (a cluster) are re-classified into sub-clusters based on structural patterns. In the first step, activity based process similarity measures are used and in the second step, transition similarity measures are used.

Qiao *et al* [133] propose a two-level business process clustering and retrieval that is more adapted for real-life business process repositories. Their approach is based on assessing both the textual (such as functional descriptions) as well as the structural (such as control and data flows) information of process models. In the first level of clustering, they apply a topic language modeling. They begin by extracting the text (such as activity labels and process documentation) and then, they convert it into a collection of documents in accordance with their text features (represented by a text feature vector for each process). For instance, all processes that have the same topic, such as "order", will be clustered together. Then, through probability analyses, the topic mixture for each process is determined and the topic with highest probability is chosen for that process. In the second level of clustering, each defined cluster (from the first level) is analyzed using graph partition approach. If the structural similarity between two nodes (representing business processes) is above a predefined level, they are connected with an edge. The connected nodes within one cluster (from the first level clustering) form the second level clustering. For retrieval, a function will, based on similarity, return the most relevant (in descending order) process models.

2.4.2 Merger of Process Models

Another set of approaches focuses on merging process models as a way of managing them. By merging process models, a reduction in overall size is achieved and as such, improves the maintainability of the collection of process models as a whole [48].

Li *et al* [102] have developed a method that merge variants into one reference model. If given a set of process variants that are similar (i.e., variants), their method will create a reference (generic) process model. This process model will be constructed in such way that the change distance (for example insert, delete or move actions) are minimal between the reference model and the similar process models it has "merged". By defining a reference model with minimal change operations needed to "become" one of the variants it subsumes, one will find the most efficient reference model (i.e. requiring less effort to configure).

The method of Li *et al* [102] cannot provide the analyst with the behavior of the input process models. Gottschalk *et al* [68] address this limitation. In their method, one can see the behavior of the input process models and also additional possible behaviors of the process. Their method, which is based on EPC, works in three phases. In the first phase, the input EPC process models are reduced to their active behavior (reduction of an EPC by removing the gateway nodes and adding them on the arcs connecting functions) and represented as functional graphs. The resulting functional graphs are then merged into a new function graph that shows the combined behavior of the input EPC models. Finally, the merged functional graph is converted back to EPC.

Reijers *et al* [137] recognize the occurrence of similar process model fragments in a repository of process models and suggest managing them by representing them as few times as feasibly possible. In order to achieve this, they propose an extension to EPC called aggregate EPC (aEPC). aEPC seek reduce the number of process models an analyst has to work with by merging two process models through combining their commonalities. In this way, identical parts of two process models, are only represented once in the repository. For instance, two similar process models for handling of a loan application (one for private and one for corporate clients), will be merged into one process model where the common parts are included only once. Reijers *et al* [137] also propose an algorithm that allows the stakeholders to easily extract the model (for example the process model for corporate clients) on demand.

La Rosa *et al* [150] propose a process model merging method allowing extraction of the input process models from the merged model. This enables the analyst to trace a certain activity to its process model and at the same time, having the behavior of the input process models subsumed by the merged process model. The algorithm first extracts the common parts of the input process models and creates a copy. Once this is done, the algorithm manages the differences by creating configurable connectors. Finally the algorithm will "clean" the process model from e.g. redundant elements.

Schunselaar *et al* [166] conducted a case study of Dutch Municipalities that offer the same services (such as registration of birth) but with similar business processes. They identified benefits of merging existing variants into configurable process models for the municipalities. However, the configurable process models must produce sound process models when being instantiated. Furthermore, they must be fully reversible, i.e., the input process variants should be instantiations of the configurable process model. They propose fulfilling the above stated requirements by introducing CoSeNet process models. CoSeNets is a tree-like block structured process models that capture the business processes. The CoSeNet process models are read from left to right with each leaf representing a task and each parent node representing an operator (sequence, logical connectors OR, AND, data-driven XOR and event-driven XOR). The parent connectors are linked through VOID nodes (linked with edges to the parent nodes). The configuration of CoSeNet process models is achieved by blocking and/or hiding VOID nodes.

Mendling and Simon [115] propose a method for merging two process models (EPC) that represent the same business process but from different views. Different views could be two EPC describing the process of receiving customer inquiry. One of the processes is from the Project Management branch and the other from Sales and Distribution branch. These two EPC represent similar processes and share common parts. The method proposed [115] consist of three steps. The first step is to identify the semantic relationship of the two input EPC process models. The process designer identifies the equivalence and the sequential order of functions and events in the two EPC models. Then, as a second step, an integrated EPC is created from the two input EPCs. This is achieved by

first creating an integrated EPC where all the elements of the two input EPCs are included. Then, nodes that capture the same thing in the process are merged into one node and the ingoing and outgoing arcs are managed with split and join connectors. The third and final step is applying a set of restructuring rules that cleans the integrated EPC model from unnecessary structures by removing redundant arcs and eliminating connectors with only one input and one output arc.

2.5 Process Model Customization

The fourth and final quadrant in this classification, envelopes approaches that seek to manage variability by extraction from consolidated process models. These approaches work with process models and rely heavily on annotations in order to extract one or more variants.

Process Model Customization approaches can be further divided into two sub-categories, behavioral and structural based approaches. Behavioral-based encompass approaches that base their method on selecting and hiding operations, i.e. selecting elements that are included in the variant being extracted and hiding elements not included. As such, there is no actual structural change to the process. This sub category includes most approaches within "process model customization" classification. In contrast to behavioral based approaches are structural-based approaches. These rely on change operations (such as insertion and deletion) for extracting variants from a consolidated process model, causing a structural change in the process models.

2.5.1 Extensions of Feature Models

Within the domain of Software product line engineering (SPLE), product variability is managed with the use of feature models. In this setting, it is important to define the commonalities and the variations of the product line being supported by an information system. The commonalities are those features that are shared by all products and variations show what features differ for different product lines. In order to manage the features of a product line, it is essential to explicitly document the variations [127].

Feature models were first introduced by Kang *et al* [33] as part of the Feature Oriented Domain Analysis (FODA). Since then, different feature modeling languages have been proposed such as [18, 32, 57, 107]. In general, a feature model consists of one or more feature diagrams that are represented as a tree-structure. At the top of the tree, high level features are depicted. Then they are decomposed into sub-features. Within a feature diagram, the features can be either mandatory or optional. Constraints among sub-features are graphically represented in a diagram. The foundational relations in a feature diagram are AND (all the sub-features must be selected), XOR (only one feature can be selected) and OR (one or several of the sub-features can be selected). An OR relationship can be specified in more detail using a n:m cardinality. In this case,

n stands for the minimum and m for the maximum number of allowed sub-features that can be chosen at a given OR relationship.

The initial aim of feature models were to aid with configuration of software product families. As such, it can only give a static view of the different features (variants) whereas a process model can capture the control and data flows between different activities. Both will represent the same case but a feature model will only capture the variability and give a structural view of the business case whereas a process model will capture the behavioral view as well [91]. However, feature models have been extended and used to provide abstraction for customization of process models in various approaches. In these approaches (see below), one can customize a process model by selecting/deselecting features from a feature model. In order to do so, one has to first establish a mapping between features on the one hand, and variants of variation points in the process model on the other hand. Once a feature configuration has been completed, an algorithm exploits this mapping to select the right variant(s) for each variation point of the process model. Then an individualization algorithm, if available, is triggered to individualize the customized process model.

2.5.1.1 PESOA

Puhlmann *et al* [132, 164] developed, within the frame of the PESOA (Process Family Engineering in Service-Oriented Applications) project, a method for improving the configuration of process-oriented software systems and managing variability in processes. The core of their approach for managing variability is to annotate process models (applied both on UML AD and BPMN models) with so called stereotypes. The activities of an UML AD or the tasks of a BPMN model are marked with <<VarPoint>> if variability can occur. An activity or task marked as variation point is abstract and has to be realized with an actual activity or task, which is, denoted <<Variant>>. It is possible to denote the default variant with <<Default>> and the other options as <<Variant>>. However, if the alternatives at a variation point are exclusive (corresponding to an XOR split in BPMN), the activity or task is denoted with the stereotype <<Abstract>> instead of <<VarPoint>>. As a shortcut, it is possible to denote the default variant of an exclusive variation point with <<Alternative>>. If an activity or task is denoted <<Null>>, it is an optional variation point and in such cases, it can only be associated with only one variant that is either chosen or not. As a shortcut, one can use <<Optional>> instead of using <<Null>> and its associated variant.

Razavian and Khosravi [134] propose additional stereotypes focusing on optional and alternative variation points. They propose denoting control-flows with <<opt_vp>> (optional variation point) and <<alt_vp>> (alternative variation point>>. Furthermore, actions can be either denoted <<optional>> or <<vp_al>>. If it is denoted <<optional>>, there is an option to choose at most one variant but if it is denoted <<vp_al>>, one variant must be chosen.

An similar approach is put forward by Kulkarni & Barat [94]. In this approach, a generic activity (called abstract activity) can be replaced by a single (atomic) concrete activity or by an entire sub process (called composite activity). Kulkani and Barat also suggest that feature diagrams can be used to guide the customization process, but they do not specify any concrete mechanism for linking a process model with a feature diagram.

2.5.1.2 Superimposed Variants

Czarnecki and Antkiewicz [57] propose a method (superimposed variants) for connecting a feature diagram with UML AD for managing variability. They state that a feature diagram is restricted to representing commonalities and variability as merely symbols. However, if features are mapped to other models that capture behavior (such as UML AD), they will be more than just symbols (they will get semantics). They propose annotating the control-flows of an UML AD with presence conditions (PCs) and meta-expressions (MEs). A PC will let you know if the element it is connected to, should be present or to be removed. MEs are used to compute attributes of model elements relevant to the UML notation (such as the name of an activity). Both PCs and MEs are captured in Boolean form over the features and its attributes of a feature diagram, and are evaluated against a feature configuration. These formulae can be represented in disjunctive normal form or as XPath expressions. UML stereotypes are used to create annotations for these formulas to be assigned to model elements. The assignment of stereotypes to modeling elements is done through rendering mechanisms such as labels, color schemes or icons. Process configuration is achieved by evaluating PCs and MEs against a feature configuration. Those model fragments where PCs evaluate to false are removed from the model, while those model attributes that are affected by MEs are changed accordingly (e.g. an activity name is changed).

Another approach that follows the same idea is that of Ripon *et al* [143] where activities in a UML activity diagram can be marked with a stereotype "variant". An activity tagged with a "variant" stereotype is linked to an entry in a variant model and a decision table. By selecting/de-selecting features, a process modeler can determine which variant of an activity will be picked during configuration time.

2.5.1.3 Feature Model Composition

In Acher *et al* [10], a process (called workflow in the paper) is defined as a collection of services, where each service corresponds to an activity. Activities are implicitly related via data dependencies. Specifically, each service has a number of data ports. A data port corresponds either to an input data object or to an output data object. When an input data port of a service refers to the same object as an output data port of another service, there exists an implicit data-

flow dependency between these services. In order to capture variability, a service is allowed to have any number of variation points (called concerns). A concern is akin to the notion of configurable function in C-EPCs, with the caveat that in the approach of Acher *et al* [10] a service may have multiple concerns, meaning that it can vary along multiple dimensions. Each concern is modeled as a separate feature model, which captures which data ports are enabled or disabled for each of the alternatives in the concern. A concern of one service may be incompatible with a concern of a second service and therefore, a consistency check is needed when customizing a workflow. Analyzing the input and output data ports of services, based on dependency rules, performs the consistency check. Specifically, the feature models of the relevant concerns are checked for mutual consistency and then a merged intersecting feature model is created. In this way, the consistency of two connected services is ensured. When producing a customized workflow, it is necessary to add the control-flow dependencies based on the implicit data-flow dependencies. Acher *et al* [10] recognize three types of control flow dependencies: sequential, concurrent and conditional. The dependency rules for consistency checks between two services are not sufficient when there is a sequential, concurrent or conditional ordering of more than two services. This is addressed by Acher *et al* [10] via a modified set of dependency rules that ensure the consistency of services in a customized workflow.

Ciuskys and Caplinskas [32] present another approach that is subsumed by Feature Model Composition. In this approach, the only process model elements that can be made customizable are activities. A configurable activity is called a generic activity. During configuration, a generic activity can be replaced by one of several possible concrete (non-generic) activities, each of which is linked to a feature. Alternatively, an activity may be skipped (removed) during configuration if it is marked as optional. The space of customization options is specified using a feature diagram, where each feature corresponds to an (generic or non-generic) activity. As is usual in feature diagrams, features are related hierarchically via XOR, AND, optional or mandatory dependencies. Also, a pair of features may be related via a "requires" relationships signifying that inclusion of one feature requires inclusion of the other feature. The features that are inner nodes in the feature diagram represent generic activities, while the leaf features correspond to non-generic activities. A process modeler configures a process model by selecting features in the feature diagram. These features then determine how the generic activities in the process model are configured, i.e. which concrete activity is selected for a given generic activity or which generic activities are removed during configuration. For the purpose of reasoning about the consistency of a given subset of features of a feature diagram, the feature diagram is translated into a theory in description logic (DL) and a "DL reasoner" is used for consistency checking.

2.5.1.4 Kobra

Atkinson *et al* [16] developed Kobra (Component-Based Application Development) method for UML 1.x and extended their method for UML 2.x [15] for producing component-based software systems. Although the main purpose of Kobra does not deal with variability, the method supports customization of process models. The Kobra method is based on considering models and modules as individual components. A product is, for example, a certain set of components where each component has its own description. A process variant can be extracted with the aid of decision models (called decision libraries that contain decisions). Each decision is associated with a certain set of possible possibilities (called resolutions). The resolutions are then linked to the variation points of a process model (in this case a UML AD) where they are marked as special splits (marked with black background) with its outgoing branches representing the paths that can be taken for a decision. It should be noted that the method does have functionality to "stop" wrong configurations or secure correctness in the configured models.

2.5.2 Questionnaire Model

The approaches presented so far (within the process model customization quadrant) require the domain experts to have familiarity with the notation language used to represent the business processes (such as UML AD or BPMN). However, if the users do not master the notation language used to annotate the models, they are of little use. La Rosa *et al* [147, 152] address this issue by introducing a questionnaire driven configuration of process models. This is achieved by capturing the variability of a process model with Boolean domain facts at each configurable node. A questionnaire model is built that is connected with the facts and the nodes. The user will answer questions by choosing from alternative responses. These responses are in turn connected with facts and nodes. Depending on the answers (Boolean in their character), the configuration is triggered by configuring the respective node with the selected alternatives and removing irrelevant paths.

Pascalau and Rath [15] propose a similar method to the questionnaire model. Their approach is an ontology-based approach to manage variations in business process models by connecting the reason for which a variation exists to its variants. It is a method of managing variations that allows the annotation of business facts (annotations on the outgoing branches of a gateway, also referred to as the reason for which a variation exists) in the process models.

2.5.3 Provop

Hallerbach *et al* [71, 72] introduce Provop (PROcess Variant by OPtions) to manage business process with large number variants. The Provop method is

based on deriving a variant of a process model by implementing a set of change operations on a base model. The base model is annotated (adjustment points) in such a way as to allow configuration. The authors propose choosing a base model based on one of five policies. It could be the standard or reference process, the most frequently used process, a process model that is the minimal average distance between itself and all its variants [102], merger of all variants in one large consolidated process model (superset of all process variants), or intersection of all process variants (the base process comprises only of elements common to all process variants. In order to extract a variant, the base process is subjected to one or several of the following adaptation patterns: Insert, delete, move or modify. Provop supports the following relations: Dependency (such as A depends on B), mutual exclusion (if A then not B) and hierarchy (managing inheritance such as if A belongs to parent B and is changed, then the parent B will also be changed accordingly). With these relations defined, Provop secures consistency.

2.5.4 Templates and Rules

Kumar and Yao [95, 96] propose capturing variability by processing a set of business rules associated with a process template. The process template is a simple, block-structured process model, which should be chosen in order to have the shortest structural distance to all process variants of a family. The rules can be used to configure the template by restricting or extending its behavior via change operations. Change operations affect the control-flow perspective (by deleting, inserting, replacing or moving a single task or a process fragment), the resource perspective (by assigning a role to a task), and the data perspective (by assigning a value to a data attribute or changing the value of a role's property or of a task's input or output data). It is also possible to change the status of a process among four predefined values (normal, expedite, urgent and OFF). Rules associate change operations with a Boolean condition over so-called case data, so that if the condition is satisfied, the corresponding change operation is applied onto the process template. Depending on the type of operation, the approach differentiates between control-flow rules, data rules, resource rules and hybrid rules (the latter incorporating multiple process perspectives).

2.6 Summary and Discussion

2.6.1 Brief Summary

This chapter began with an overview of approaches to manage variability in business processes and process models. It did so, with the aid of a framework for classifying the approaches according to two classification parameters. The first parameter concerned if the approach proposes an extension or reduction of the number of business processes or process models. The second criteria evalu-

ated if the variability was managed as part of creating or modifying actual business processes or process models representing business processes. By combining these two classification criteria, a framework for classifying approaches, consisting of four categories or quadrants, was created. The quadrants are "*business process standardization*", "*business process customization*", "*process model standardization*" and "*process model customization*". The identified relevant approaches were classified and briefly described.

The classification encompassed a total of 35 different approaches. Most of them are classified as "process model customization" (12 approaches), "process model standardization" (9 approaches), "Business Process Customization" (8 approaches) and finally, "business process standardization" (5 approaches).

2.6.2 Observations

The review revealed some commonalities and differences between the quadrants of approaches. These observations concern input consistency, output correctness, flexibility in implementation and root causes of variability.

2.6.2.1 Input Consistency

Input consistency considers how consistent the input has to be for the approach to function. The input consistency considers the required consistency of the notational language the input models are represented with. For instance, an automated approach that applies an algorithm on a set of process models will require a high degree of notational consistency for it to be successful. Such an approach would yield very limited results if the set of input process models would include different notational languages (such as some in EPC and others in BPMN), have inconsistency in its annotations of elements (not complete as in lacking in some parts), or have inconsistent labels on its elements (such as same task having two different labels).

In addition, input consistency, also relates to the consistency in the hierarchy of the input process models, requiring they need to be of equivalent level of decomposition. For instance, if an approach can be applied on a set of process models that are not on the same level of granularity (such as some processes are only modeled as a sub-process whereas other are decomposed to the lowest level of granularity), it does not require hierarchal consistency.

If an approach requires consistency in terms of notation and decomposition of the input process models, the input consistency is "high". Conversely, "low" input consistency means that the approach will work even if the input models are not consistent in regards to notation language or level of granularity.

2.6.2.2 Meaningful Variant Output

Meaningful variant output considers if the approach ensures that the variants produced, are meaningful, i.e. the degree to which the models reflect the business reality and is considered by domain experts to be a meaningful representation of the variants. As such, this does not refer to syntactic (correct structure of the process model such as avoiding disconnected nodes [50]) or semantic (correct behavior of the process model such as ensuring that there are no deadlocks [50]) correctness. For instance, process model customization includes existing variants of a process and, assuming correct annotation, the output of these approaches will result in variants that make sense to the domain experts. However, approaches that are classified as process model standardization could, for instance merge two process models that are not considered as variants of each other (based on e.g. similarity). Such approaches would require domain experts' assessment or approval. If an approach requires assessment from domain experts, the output models can be considered to be "low" as compared to if is not required ("high").

2.6.2.3 Flexibility in Implementation

Flexibility in Implementation evaluates the degree of flexibility when implementing approaches. The relevancy of this observation is due to the varying contexts of each project. Each context requires different degrees of flexibility when implementing an approach. For instance, during "business process standardization" when new processes are being designed, higher degree of flexibility is needed. Conversely, if the project is more geared towards understanding existing variants (process model customization), it is more important to retrieve correct results within an acceptable timeframe. As such, the degree of flexibility is not as relevant as when creating new business processes. If the implementation of an approach requires adhering strictly to given instructions, it is considered to have "low" flexibility. However, if the instructions are generic, allowing for changes and adaptations during its implementation, the flexibility is "high".

2.6.2.4 Root Causes of Variability

All variations in business processes, and therefore in also in the process models representing the business processes have a root cause of variability. This observation concerns if the root cause of variability is considered or captured in the approach. In other words, if an approach uses the cause of variability as a parameter in its process modeling efforts, it is considered to include the root cause of variability. As such, approaches that disregard, implicitly or marginally considers the business reason for variation, have "low" connection between the model and the business logic behind the variability. On the other hand, approaches that consider the root cause of variability and use it as an important

input, have “high” connection between the model and the business logic behind the variability

2.6.2.5 Summary of Observations Made

The observations made are summarized in the Table 1. The first column lists the quadrant of approaches. Each of the following columns shows the observations as described above.

Table 1: Summary of Observations

Quadrant	**Input Consistency**	**Meaningful Variant Output**	**Flexibility in Implementation**	**Root Cause of Variability**
Business Process Standardization	Low	High	High	Low
Business Process Customization	High	High	High	Low
Process Model Standardization	High	Low	Low	Low
Process Model Customization	High	High	Low	Low

All quadrant with the exception of business process standardization, require high degree of input consistency. Such approaches require that the models use the same notation, at the same level of granularity and using the same modeling conventions and vocabulary. If these prerequisites are not fulfilled, the input models need to be “translated” into required notation language. This requires considerable effort on part of the process modelers. However, it should be borne in mind that even approaches that do not require consistent input, such as business process standardization approaches, still require considerable effort.

The required effort to validate the output process models is high for all quadrants with the exception of process model standardization. These approaches apply automated methods on a set of process models and produce an outcome as according o a set of rules predefined in the algorithms. As such, there is no insurance that they are “meaningful”. As such, the resulting models from process model standardization need to be reviewed by domain experts in order to ascertain their meaningfulness. This could prove to be a time consuming activity. For

the other quadrants, the models created as output, have been reviewed concurrently (business process standardization and customization) or prior to the modeling work (process model customization).

When working with improving or creating new business processes (via the aid of process models), flexibility in implementation is given at the cost of correctness. However, once the business processes have been captured and they need to be improved (as models), consistency and correctness is valued higher. As such, approaches dealing with business processes have high flexibility in implementation whereas approaches working with models have low flexibility in implementation.

The main (in majority of cases, the only) parameter for managing variability, is syntactic similarity across process models. Consider for example the algorithm for merger of process models [150]. This algorithm considers the syntactical differences between two input process models and will create a merged model that subsumes the input process models. As these approaches do not do not take into account business reasons for variability, there is a risk of creating overly large and complex process models where variants are merged when it does not make sense to do so from a business perspective. When the business reasons for variations are not actively considered, it can result in a discrepancy between process models and the reality they are representing. The greater this discrepancy becomes, the more unfamiliar will they become in the eyes of the domain experts. Furthermore, the models will have limited value for instance as a tool for business improvements. As such, all quadrates have “low” consideration for the business logic of variations when modeling.

In conclusion, despite the existence of many approaches to manage variability in business process models, they have the shortcomings of (1) operating on the assumption of consistency and (2) disregarding the reason for variability in determining how to structure the process models. The assumptions of consistency are unrealistic in many practical scenarios where models of each variant might not be available to start with, and even if they were available, they would typically have been modeled by different teams and using different conventions. The models are probably also not modeled consistently at the same level of granularity, i.e., some models are at a detailed level whereas others at a higher level of granularity and only represented as a sub-process.

The next chapter presents the foundations of process decomposition, which constitutes as one of the two pillars of the systematic method of this thesis for managing variability that address the shortcomings observed above.

3 FOUNDATIONS OF PROCESS DECOMPOSITION

Business process models [155, 190] are used for a wide range of purposes, ranging from internal communication and knowledge management, to process improvement and information systems requirements engineering [19]. Given this multifunctional character, process models need to be captured in a way that facilitates understanding and maintenance by a variety of stakeholders. In this respect, it is generally accepted that large process models should be decomposed into smaller sub-processes.

Decomposition methods rely on the principle that a larger complex phenomenon is more easily understood and grasped if it is broken down into smaller parts. This concept has been used in other domains within the field of computer science such as systems theory or conceptual modeling. Within system theory, for instance, decomposition is defined as "breaking down a complex system into smaller, relatively independent units" [123]. Within the context of business process modeling, decomposition (a.k.a. modularization [138]) is referred to when a section of a process model is clustered together as a sub-process.

A process model can be decomposed along two lines, vertical and horizontal. A vertical decomposition improves understandability and maintainability by obtaining process architecture (hierarchy of processes and sub-processes without their details). Horizontal decomposition, on the other hand, improves understandability and maintainability of a business process by structuring the models in manageable chunks, the output of which is a set of process models, related by means of "parent-child" relations. As such, the horizontal decomposition enhances the vertical one by adding the control-flow relations linking sub-processes at a given level of hierarchy and adds the details to each process in the hierarchy.

This chapter is structured as follows. Section 3.1 discusses vertical decomposition and 3.2 focuses on horizontal decomposition of process models by reviewing existing literature on decomposition heuristics and metrics. In Section 3.3, the effect of applying decomposition heuristics on a flat process model is examined with a case study. Section 3.4 examines the same effect but with an experiment and finally, Section 3.5 summarizes the chapter.

3.1 Vertical Decomposition of Process Models

A number of methods for vertical process decomposition exist within the field of process modeling. Although these methods differ in terms of the nomenclature, specific definitions (for instance about various levels of the process decomposition), and how they approach the task of decomposing business processes, they rely on a common set of core concepts that are summarized below.

At the highest level of process decomposition, the business process map, defined as a "*conceptual model that shows the processes of a company and makes*

their relationships explicit" [54] , is represented (cf. Fig. 2). These processes can be categorized in different ways. Rummler and Brache [155] suggest a division of the processes in either core processes, support processes or management processes [155]. Sharp and McDermott [167] and Porter [131] suggest only two categories, core processes and support processes. Regardless of how they are categorized, all make a distinct difference between those processes that serve external customers and are very closely aligned to why the business exists. These processes are called core processes such as marketing & sales or operations. All other processes are considered to be support processes such as procurement or human resource management. These support processes all share the commonality of serving an internal customer.

These processes (core and support), such as procurement, have a *main process* (cf. Fig. 2). A main process is a process that does not belong to any larger process (but might very well be categorized as for instance a core process). The main process is decomposed into a number of *sub-processes* or "diced" [58] where each sub-process represent a high-level activity needed to fulfill the main process. For instance, in Fig. 2, the main process for procurement is "diced" into four separate sub-processes, namely, prepare purchase order, order materials, receive goods and pay invoices. Sub-processes are processes on their own and it can be further decomposed into sub-processes until such a level where a sub-process consists exclusively of atomic activities (called tasks) that do not warrant further decomposition (cf. Fig. 2).

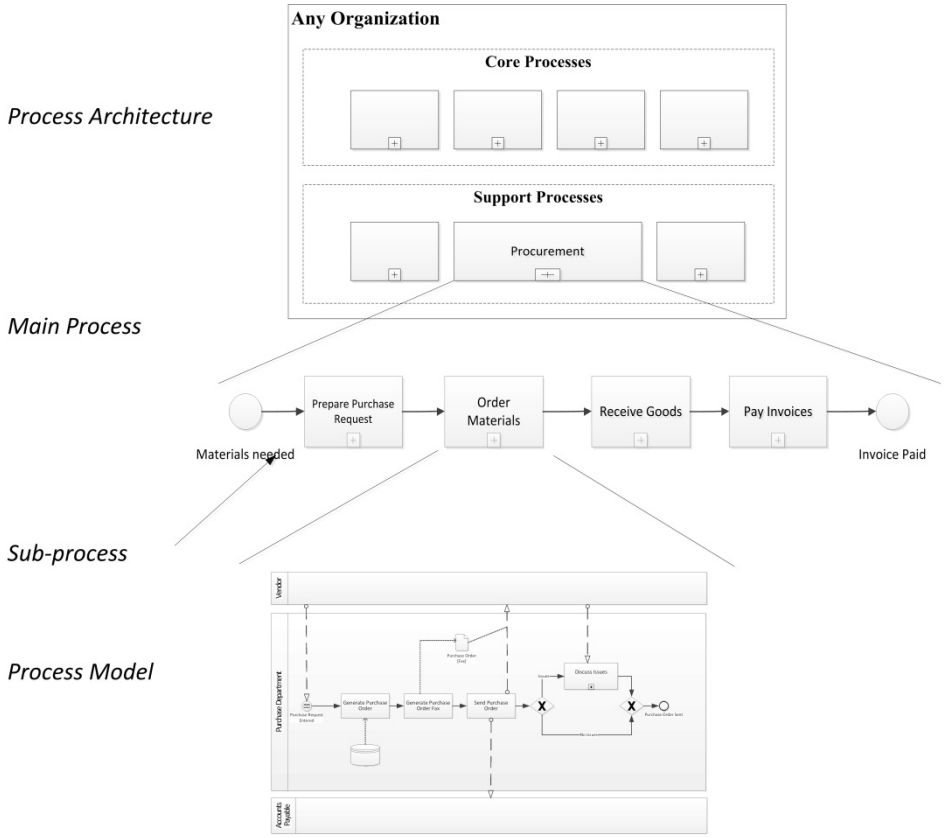


Figure 2: Illustration of Process Architecture

All of the levels describe above, make the business process architecture which is defined as "a hierarchical structure of process description levels and directly related views covering the whole organization from a business process point of view" [40]. As can be seen from Fig. 2, a process architecture starts with a high level process map and continues down to a detailed process model describing specific tasks, roles, IT systems and so on. A process architecture should consist of four [40] or five [155] levels but can have more levels when necessary. In this thesis, high level of decomposition refer to level 2 and 3 (level 1 refer to the process map) of the process architecture proposed by Rummler and Brache [155]. Using the same process architecture, low levels of decomposition refer to levels 4 and 5 (where 5 is the lowest levels of decomposition).

Note that the above discussion refers to business processes, regardless of how they are represented. When modeling a business process, however, it is only natural to model each of its sub-processes separately. Accordingly, the hierarchy of processes derived via process decomposition is reflected in a corresponding hierarchy of process models representing the sub-processes in this decomposition. It is noteworthy that literature on vertical decomposition does

not consider (by providing guidelines or instructions) variability. As such, vertical decomposition is "merely" representing the sub-process with higher level of detail by "dicing" or "chunking" the sub-process into smaller pieces. Therefore, vertical decomposition is in essence the same as horizontal decomposition. The sub-processes are therefore only "diced" but not "sliced" when decomposed.

3.2 Horizontal Decomposition of Process Models

The benefits of horizontal process decomposition are acknowledged [81] but there is far less consensus as to how a given process model should be decomposed [138]. Several guidelines and goodness criteria for process decomposition co-exist. There is also a lack of evidential comparison of their relative merits. For instance, some authors propose that goodness of a decomposition should be assessed based on size [114, 194], while others suggest transposing modularization criteria from information systems [81, 139]. Some propose to decompose processes based on data [78] while others propose role-based decomposition [86]. Despite the plethora of available approaches, some maintain that decomposition is more an art than it is a science [27].

In this setting, this section of the thesis addresses two research questions: (1) "How can process models be decomposed?" and (2) "How do different decomposition approaches affect a process model in terms of metrics associated with maintainability and understandability?" The first question is addressed via a literature review and classification of process model decomposition approaches. The second is addressed via an empirical case-based evaluation and an experiment. Specifically, two representative decomposition heuristics are applied on a real-life process model that was originally modeled flat from start to end. Then the resulting models are compared using a range of maintainability and understandability metrics.

3.2.1 Literature Review

The literature search process was based on the principles given in [88]. Queries were submitted to Google Scholar (which encompasses relevant databases such as ACM DL and IEEE Xplore). Three different advanced searches were applied. The following keywords were used in these three queries; "modularization" OR "decomposition" OR "sub-process" OR "fragment" OR "abstraction" OR "refactoring" in separate combinations with (AND) the keywords "process model", "process modeling" and "workflow", and gathered the first 400 hits of each of the three queries (1200 hits in total). The search was conducted in October 2014.

In the first round of filtering, based on title only, duplicates were eliminated and papers that were clearly off-topic were eliminated as well. After this iteration, the list shortened to 177 candidate papers. The following step was an inspection of the abstract and the introduction of each paper so as to eliminate

papers that did not deal with process modeling. For instance, many papers deal with decomposition or modularization of information systems or software code but had no significant relation to process models. Papers that did not propose a specific approaches to process decomposition, but instead dealt with another topic and referred to process decomposition as a separate issue, were also excluded. At the end, the list contained 67 relevant publications.

An initial analysis of these 67 publications revealed two distinct categories. On the one hand, one subset of publications (50) provided prescriptive methods or guidelines for decomposing a given process model into sub-process. The other subset of the publications (17) proposed *criteria* and associated *metrics* to assess the "goodness" of a given decomposition without prescribing how a process model should be decomposed in order to achieve a suitable level of goodness. Herein, the term *decomposition heuristics* is used to refer to approaches in the first category and *decomposition criteria* to refer to the second category. Below we discuss each category in turn.

3.2.2 Decomposition Heuristics

The 50 publications dealing with decomposition heuristics proposed, as expected "manual" methods or guidelines for process model decomposition. However there were papers with decomposition heuristics being employed in the context of process model abstraction, refactoring, architecture and automated model fragment extraction. These are summarized and presented below.

A process model can be decomposed based on "milestones" in the process. For instance, when decomposing existing EPC models, Davis [39] proposes to look for those parts in the process where there are (1) limited connection to other parts of the process, (2) connected events, (3) limited use of loops and (4) a common distinct theme (such as order fulfillment). Sharp and McDermott [167] adopt a similar approach where they seek to decompose a process at those points in which significant milestones in the overall process is achieved and usually are points of interest in terms of process measurement.

If the activities of a process model are annotated with *data objects*, the process model can be decomposed based on the relation between the activities and the data objects. For instance, Ivanovic *et al* [78] propose fragmenting a workflow, based on the data objects, by looking at what and how many data inputs an activity has and which other activities share the same data objects. They postulate that if many activities use the same data objects, they are related and thus belong in one process fragment.

Decomposition based on stakeholders, i.e. who or which resource is performing the activities, is another broad decomposition approach. For instance, for business process outsourcing, it has been proposed in [86] that fragments where the activities, which a given stakeholder or partner is responsible for or is the object of outsourcing, are modeled together as one sub-process. Another example is when several stakeholders are engaged in modeling a larger process

model. For such purposes, it has been proposed [87] to have each stakeholder model their partial models (sub-processes) which are then used as sub-processes of a complete process model. A similar approach [56] considers that knowledge is fragmented at the local level and therefore, these fragments can be the building blocks for a larger process model. As such, each modeler with local expertise, models their fragment (sub-process) that is subsequently put together with other fragments. Subject-oriented BPM (S-BPM) sets the subject of the process in focus where decomposition (subprocess) captures the activities performed by a specific role [178].

Another approach when modeling processes is based on its goals and sub-goals of a process. Goal modeling has its roots in requirement engineering and is used to define the objectives that a system should achieve [126]. In the context of business process modeling, goal-based approaches, such as [14, 93], determine the decomposition of a business process based on its goals and sub-goals. Specifically, elements in the process (e.g. activities) are clustered based on the goals/sub-goals they intend to achieve, and the resulting clusters are mapped to separate sub-processes.

Product development processes are those that transform a technical solution to a product that can be delivered to customers [191]. A set of modeling approaches proposed for this type of processes relies on design structure matrices as input for decomposition of the model. For instance, some studies [59, 97, 145] propose decomposing the activities (that are assumed to have an output/input relationship) based on their iterative characteristics. In such approaches, the design structure matrix helps identify those sets of activities that are sequential, parallel or cyclical and these characteristics are used as basis for the decomposition. Other heuristics of decomposition in product development processes are based on subjective assessment of the strength of the interactions between tasks.

Abstraction: Business Process Model Abstraction recognizes that different stakeholders are interested in seeing different levels of process model detail. Abstraction methods cater to this need by applying techniques on detailed process models, resulting in a generalized version of the same process model [130]. As such, abstraction techniques collect and cluster a certain set of atomic activities or sub-processes and represent them with one aggregated sub-process. A common denominator of abstraction techniques is to first determine the selection criteria for significant activities, such as roles (resources), activity frequency or activity completion time [101, 128, 170], structural aspects of a process model [129] and semantic aspects [158, 169]. Once the perspective is chosen, the process models are transformed accordingly.

Refactoring of business process models aims at improving understandability and maintainability without changing their execution semantics [46]. A process model can be refactored (i.e. changing existing sub-processes or introducing new ones) and as such, refactoring is related to decomposition. For instance, if two sub-processes or sets of activities have the same, partially the same or similar activities and flow, they can be replaced with a shared sub-process [46]. As

such, the criterion of fragment similarity is used to decompose. Weber *et al* [187] present eight process model smells and eleven refactoring techniques to address the smells. Of these eleven, four are related to decomposition of process models. The first one is related to redundancy in process models (repetition of the same fragments). In such cases, a process fragment is extracted and put as a sub-process. The second relevant technique deals with lazy process models (sub-processes containing few activities) that are consolidated to fewer sub-processes. The final two refactoring techniques aim at addressing frequently occurring deviations from the main process. These can be managed by representing them with one or more "generalizing" sub-processes.

Process model decomposition (also refereed as vertical decomposition in chapter 3) is also relevant within the field of process architecture, i.e. how the entire collection of an enterprise's process models are organized [92]. In process model architectures, decomposition is found either in the form of aggregation ("part-of relation", meaning that a process is decomposed into fully contained sub-processes) or generalization ("is-a relation", meaning that a process is decomposed into variants representing alternative ways of performing the process) [118]. Process architecture provides, in regard to process model decomposition, guidelines that are open to interpretation. For instance, zur Muehlen [118] propose primarily milestone and role based decomposition of process models. Malinova *et al* [92] found that practitioners decompose their process models based on number of elements (size), complexity or stakeholders. Dijkman *et al* [49], when investigating the prevailing process architecture designs, elicited a classification of five different structures, and thereby principles by which process models are decomposed. The main principles enlisted for decomposing process models are (1) goal-oriented, (2) function-based, (3) reference model-based (adapting an industry reference model), (4) object-based, and, finally, (5) based on business units.

Vanhatalo *et al* [185] propose a method to "parse" a process model into a hierarchy of "single entry single exit" (SESE) fragments. Since such fragments only have one entry and one exit, a change in one fragment is locally confined and thus the fragments are independent of each other. These properties entail that each fragment obtained via this method can be directly extracted as a separate sub-process, thus providing a basis for automated process model decomposition [138, 158, 169, 179, 180]. For instance, Reijers *et al* [138] builds on top of SESE extraction techniques and uses label similarity in order to determine which SESE fragments should be extracted as sub-processes. This automated decomposition method is based on the assumption that nodes with similar labels (excluding control nodes) are more likely to belong in the same sub-process than those with different labels (measured e.g. via string-edit distance).

3.2.2.1 Categorization of Decomposition Heuristics

When examining the decomposition criteria used by the methods reviewed above, 6 classes of decomposition heuristics are distinguished: breakpoints, data objects, roles, repetition, sharing and structuredness (cf. Table 2).

The common denominator of approaches subsumed by *breakpoints* is that the decomposition heuristics is based on milestones or natural breakpoints of the process. In these methods, decomposition is made at points representing natural phases of the process towards the fulfillment of its objective. For instance, heuristics based on goal-decomposition [14, 93] cut the process at points where sub-goals are achieved. Similarly other authors [39, 118, 167] propose to decompose at points where two sub-processes have distinct themes and therefore are logical milestones or separate functions in the process. Logical breakpoints are also used for decomposition in the context of reference process models, for example in the MIT process handbook [105]).

Table 2: Heuristics for Process Model Decomposition

Decomposition Heuristics	**References**
Breakpoints	[14, 39, 49, 93, 118, 167, 172]
Data Objects	[34, 49, 78]
Role	[49, 56, 86, 87, 92, 118, 125, 170]
Shared Processes	[59, 97, 145, 187]
Repetition	[46, 180, 187]
Structuredness	[138, 158, 169]

Object-based heuristics assume that activities sharing common objects belong together and thus should be in one sub-process. These approaches [34, 78] consider the objects as primary driver for decomposition decisions. Empirical studies [49] have shown this principle being widely used in the industry.

Role based heuristics ground their decomposition decisions on "who" is performing the activities [118, 125, 170]. These approaches are applied in particular to collaborative process modeling where different organizations or business units contribute with their own fragments, as proposed by [56, 87], or when modeling for outsourcing purposes [86].

Refactoring heuristics, such as [46, 180, 187] seek to reduce redundancy stemming when a process fragment is called upon multiple times in different parts of a process, i.e. *shared processes*. These shared fragments are then modeled as a sub-process.

Repetition-based heuristics look at occurrences of a certain fragment of a process. For instance, some authors [59, 97, 145] consider the frequency of sets of activities. Those sets of activities that are repeated more often (cyclical) are separated from those that are sequential or parallel. In refactoring, fragments that have frequently-occurring instance and variant changes are generalized and modeled as separate sub-processes [187]. The common denominator for these heuristics is that "occurrence frequency" is used to determine which fragments should be included in a sub-process.

The final class of heuristics for decomposition is based on the *structuredness* of the process models. The common denominator of this class is using SESE fragments as a basis for identifying candidate sub-processes as in [138, 158, 169, 179, 180].

3.2.3 Decomposition Criteria

In this section, a review and classification of those publications that propose decomposition criteria and associated metrics rather than specific heuristics, is presented.

Size metrics: It has been shown that larger process models tend to hamper understandability [114, 194] and increase the probability of making errors [113]. On this basis, "good" sub-processes are neither to small (lazy process models [187]) or overly large. For instance, the IDEF0 method provides guidance to limit the number of functions (activities) to 4 – 6 per model [118]. Others state that the number of activities should be between 5–15 [82], 5–7 activities [167] or that the number of elements should not exceed 50 [113]. Other size-related metrics proposed [29], consider number of activities, number of activities and control-flow elements, or number of activities, joins, and splits in a process model. However they do not provide further guidelines as to what, in terms of values, constitute a "good" size of a sub-process. In summary, the following the following size-related metrics are identified: number of activities and number of nodes.

Complexity Metrics: The underlying assumption for process model complexity metrics is that overly complex process models reduce their understandability and increase the probability of errors [29]. Empirical studies indicate that the number of arcs in a process model influence their understandability [194]. For measuring complexity, the Control-Flow Complexity metrics (CFC) has been proposed [28]. The CFC adds the number of branches for all split constructs of a process model. Inspired by Halstead, the "Halstead-based Process Complexity" (HPC) is proposed in [29], which measures the length, volume and difficulty of a process model. Other complexity metric proposed are CNC (coefficient of network complexity) [100] and CI (complexity index). CNC measures the number of arcs divided by the sum of all activities, joins and splits). The CI is defined as the number of node reductions that are required to reduce a process model to one node. In addition, the density of a process model is measured as an

approximation of its complexity. Density captures the relation between nodes and arcs and is defined as the total number of arcs divided by maximum possible number of arcs for the nodes of a sub-process. A high density value indicates a more complex process model and is negatively related to understandability [184].

Coupling and Cohesion: Coupling and cohesion metrics for process models have been borrowed from the field of software engineering, where they have been proposed as proxies for maintainability of software designs. Several variations of coupling measurements have been proposed in the field of business process models, including "density metrics" [112], "cross-connectivity metric" [184], connectedness" [138], "weighted coupling metric" [182], "process coupling" [139, 183] and adaptation of coupling metrics for eEPC models [24]. A common feature of these metrics is that they look at the connectedness of the control-flow elements of a sub-process As such, if a collection of nodes are connected to each other, they are more likely to be related to each other and thus should belong in the same sub-process.

Closely related to the notion of coupling is that of cohesion. Cohesion refer to how much the sub-elements of a given module (in this case a "sub-process"), are internally connected. In seeking to evaluate "good" EPC models, Daneva *et al* [35] propose metrics for "functional cohesion" that measure the level of intensity of the control flows that a given function handles; "event cohesion" that measures how much events cause complex control structures in a model; and "cohesion of a logical connector" that quantifies the weighted sum of functions, events and connectors. A cohesion metrics has also been developed based on the "steps" composing an activity and their associated data objects [136, 139, 183]. In this latter case, "relation" and "information" cohesion of an activity is measured. Relation cohesion defines at how much the steps of one activity are connected between them, while information cohesion measures how many information elements are used more than once in relation to all information elements. While this metric focuses on one individual activity at a time and not on an entire sub-process, it is possible to lift this cohesion metric to sub-processes by calculating the average cohesion of activities in the sub-process.

Table 3: Process Model Decomposition Metrics

Decomposition Metrics	**References**
Size	[29, 82, 118, 167]
Coupling and Cohesion	[24, 35, 112, 136, 138, 139, 182–184]
Complexity	[28, 29, 100]

3.2.4 Discussion

The review show that some decomposition heuristics can be applied surgically, i.e., on sections of process models that exhibit specific patterns. The "repetition", "shared processes" and "role based" heuristics are possible to implement on process fragments if and only if certain conditions are fulfilled. For instance "repetition" and "shared processes" heuristics are only applicable to process fragments that exhibit such patterns. Likewise, "role based" heuristics offer guidelines for cases with several stakeholders but cannot be applied when, for instance, the process of one stakeholder is large and needs further decomposition. As such, these heuristics do not provide enough support for being applied on a set of process models but rather function as complementary to other heuristics. They can be surgically applied when prerequisite conditions are fulfilled. The "breakpoint", "data object" and "structuredness" heuristics, on the other hand, can be applied generally on process models, as they are not dependent upon certain conditions being fulfilled.

Furthermore, we note that the heuristics do not provide sufficient criteria for determining which fragment of the process model to include as a separate sub-process. A heuristic might offer necessary criteria (statements on how to decompose) but not sufficient criteria (statements determining which process fragments to include in a sub-process). For instance, "structuredness" provides necessary criteria (SESE blocks) but does not provide sufficient criteria for which SESE fragment to use (usually there are many such fragments in a sub-process). Furthermore, "structuredness" is particularly problematic as it only states that the sub-processes should be SESE fragments but no further parameters to follow. As such, "structuredness" needs a value such as size or other criteria for "good decomposition" as discussed above. If such metrics are used, the process models will be decomposed using arithmetic that probably will not reflect the actual business processes.

In similar manner, "breakpoint" and "data object" provide necessary criteria but sufficient criteria for determining which fragments represent a "milestone step" or share same set of data objects is not defined. In contrast to "structuredness", these heuristics, while not offering sufficient criteria, can be applied by relying on the knowledge of the domain experts and should, intuitively, produce process models that better reflect the actual business processes. It should be noted that the data object heuristics requires that the data objects are modeled in a consistent way across the process models. If the data objects are not captured, this heuristics cannot be applied. Furthermore, if they are not captured consistently, the data object heuristics will be insufficient for those fragments that lack proper modeling of data objects. Finally, it should also be noted that domain experts are most likely better acquainted with the flow of their processes rather than the objects used and produced. As such, domain experts would prefer to reason along breakpoints. Nevertheless, there are currently no heuristics that provide both necessary and sufficient criteria for process model decomposition.

It is worth mentioning that the heuristics for decomposition is highly inspired by research from conceptual modeling while metrics for assessing decompositions are direct transposition of metrics from programming and software design to process models [29, 119]. For instance, coupling and cohesion metrics are inspired by Wand and Weber [81], size from lines of code (LOC) [29], modularity from information flow by Henry and Kafura [29] and complexity from McCabe's cyclomatic complexity metrics [119]. As such, decomposition heuristics and metrics have emerged quite independently of each other as separate streams of research. In the literature on process model metrics, there are no substantiated claims that a certain metric is more suited for use together with a certain decomposition heuristics. For instance, there are no claims stating that density metrics are better suited for process models decomposed based on structuredness or breakpoints.

3.3 Case Study

Case studies are often used for exploratory purposes, but they are also suitable for testing a hypothesis in a confirmatory study [60, 156] or to evaluate a method within the software and systems engineering domain [89]. These features make the case study method applicable for the purpose of this thesis, as, it is not known a priori, how the different heuristics affect a decomposition of a process model from a measurable perspective (metrics).

3.3.1 Design

The case study starts from the research question: "How do different decomposition heuristics affect a process model in terms of metrics associated with maintainability (coupling and cohesion) and understandability (size and complexity)?" Given that different heuristics should produce different decompositions, the hypothesis is that "different decomposition heuristics will result in different partitioning of model elements, coupling and cohesion, size and complexity for comparable sub-processes". It is not expected, (null hypothesis) that "different decomposition heuristics do not result in different partitioning of model elements, coupling and cohesion, size and complexity for comparable sub-processes."

The setting of the case study is the fixed income operations of a mid-sized European bank. Fixed income products are investments that give a regular and scheduled variable amount of money to investors. Examples of fixed income products are government bonds, T-bills and mortgage bonds. The original process model used in this case study was modeled for documentation purposes by a team of consultants. This model is flat i.e. no decomposition was made and as such, it begins with a start event and continues until the end of the process (including data objects and resources).

The case study consists of two main steps: (1) decompose the input process model using different heuristics; (2) compare the resulting decomposed process models. The first step is to decompose the flat process model using the breakpoint-based and the data object-based heuristics. These heuristics were chosen generally applicable on a set of process models (see previous section). The chosen heuristics were breakpoint and data object. Structuredness was not applied, as there is no guidance of how to define the size of a SESE block. The decomposition of the flat process model was conducted by two researchers, of which one is the author of this thesis. First, each author independently decomposed the flat process using each of the two heuristics. In order to reduce learning effects, the authors began the decomposition using different heuristics (one author started with breakpoint-based then data object-based and the other did the vice-versa). When both had concluded their decompositions, results were compared, discussed and harmonized, leading to one decomposed process model for each heuristic.

In the second step of the case study, a set of process model metrics were applied to the two sets of decomposed process models. For this comparison, the calculations included size, coupling, cohesion, CNC, Density, Jaccard Index of comparable sub-processes, differences between sub-processes, and their hierarchy in terms of number of sub-processes at different levels of decomposition.

3.3.2 Findings

In this Section, the results of the metrics used to compare the two sets of process models (breakpoint and data object-based decomposition) are presented and discussed.

Structure and Hierarchy: The breakpoint-based heuristics resulted in a total number of 18 sub-processes whereas the data object-based has 15 sub-processes. The two sets have 6 identical (in terms of activities inside the sub-processes) and 7 equivalent (exist in both sets covering same "function" but differ in terms of activities) sub-processes.

The hierarchy of the two sets has subtle but noticeable differences. For instance, the number of sub-processes at the highest level (the value chain) is 5 versus 6 for breakpoint and data object-based heuristics respectively. The data object-based set has an additional "settle trade payment" sub-process. Using a breakpoint heuristics, this sub-process is included in the "settle trade" sub-process. As such, there is a difference in the distribution of sub-processes at the second level of decomposition. Both sets have 8 sub-processes in total at the second level but in the data-based, two of them are encapsulated by "settle trade payment" whereas the same sub-processes are in "settle trade" in the breakpoint-based set. As such, "settle trade" and "settle trade payment" in data object-based have to sub-processes each. However the breakpoint-based set has the same 4 sub-processes in "settle trade" (cf. Fig. 3 and 4).

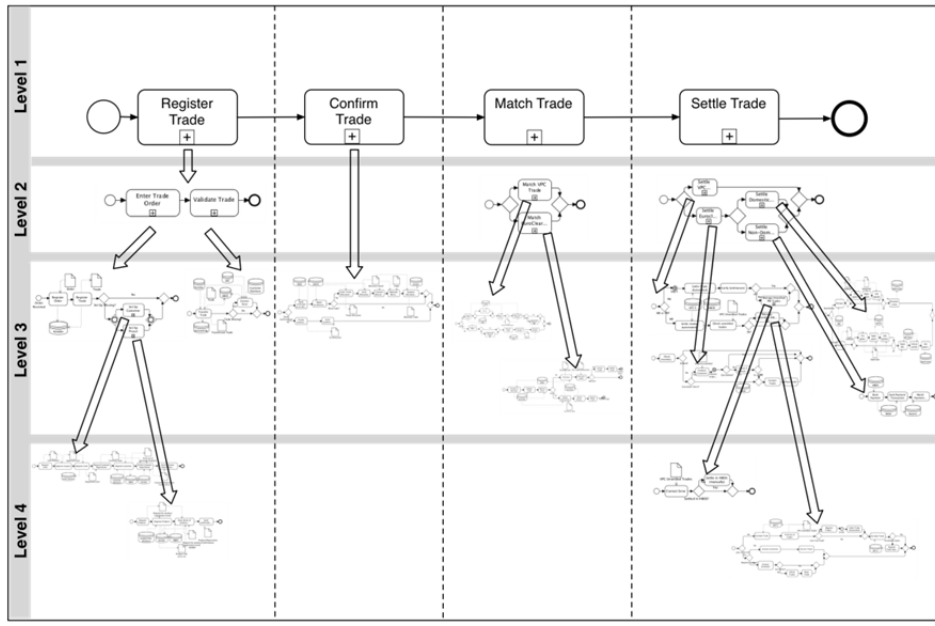


Figure 3: Process Hierarchy with Breakpoint Heuristics (readability not intended)

Another difference is within the process called "match type A/B trade" (belonging to sub-process "match trade"). With a data object-based decomposition, this process is modeled as one sub-process as its activities use the same set of data objects. However, with a breakpoint-based decomposition, a nested sub-process is introduced for managing unmatched trades. As such, there is another level of decomposition in the breakpoint-based set as compared to the data object-based set. The same case occurs for "settle type A/B trade" (belonging to "settle trade"). These two examples illustrate one primary difference when applying breakpoint versus data object-based heuristics. For breakpoint-based decompositions, nested sub-processes are used whereas it is not used to the same extent for data object-based decomposition (as the nested sub-processes commonly use the same set of data objects).

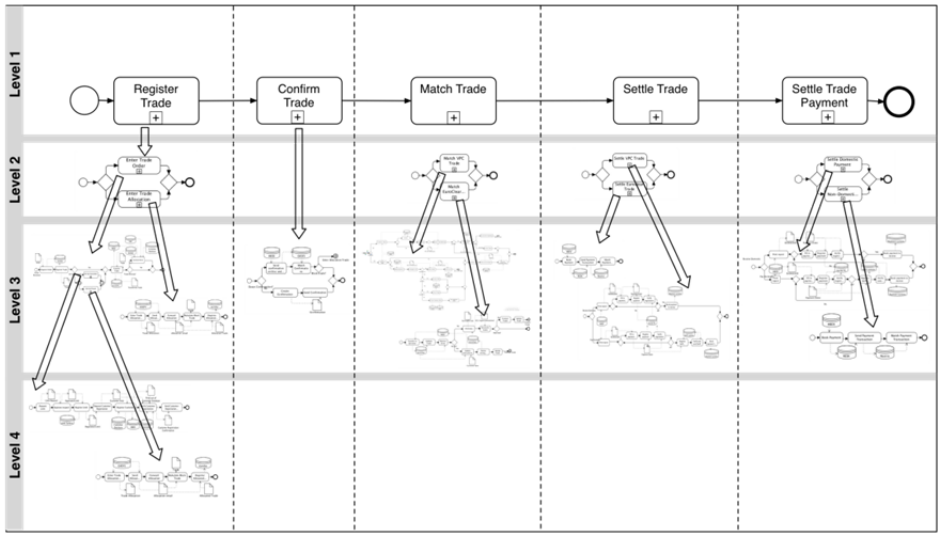


Figure 4: Process Hierarchy with Data Object Heuristics (readability not intended)

If only the sub-processes at the lowest level of granularity are considered, i.e. sub-processes that are not decomposed further, there are 15 (breakpoint-based) versus 11 (data object-based) sub-processes. Of these, 6 sub-processes were identical (Jaccard index of 1). Two of these, "enter trade order" and "confirm trade" were different. The similarity of "enter trade order" was 67% (according to Jaccard index). This is purely due to different decomposition heuristics. From a breakpoint perspective, entering a trade, and validating a trade are two distinct sets of activities or functions and are therefore separated and modeled as two sub-processes. However, from a data object perspective, the activities for entering and validating a trade use the same data objects and as such, are modeled as one sub-process. The same applies to "confirm trade" that has a similarity of only 44% (using Jaccard Index). In "confirm trade", there is an option to make a trade and allocate it between different sub-units. For instance, a counterpart might buy government bonds for 100 million EUR but then wish to distribute it to five of its sub-units, each worth 20 million EUR. As such, they will make a trade and then allocate the trade to the different sub-units. From a breakpoint perspective, the trade of 100 million is performed and under "confirm trade" it is allocated. As such, "confirm trade" include activities related to allocation of the trade. However, the data objects used for allocating the trades are not similar to those used for other activities for confirming a trade. As such, a data object-based decomposition will separate the activities related to allocation and put them in a separate sub-process.

Finally, the breakpoint-based decomposition set includes 5 sub-processes that are not in the data object-based set. Similarly, the data object-based set has one sub-process that does not exist in the breakpoint-based set. Of those sub-processes that are in the breakpoint-based set but not in the other, 4 related to

nested sub-processes discussed above (managing mismatch or unsettled trades). The fifth one, "validate trade" is related to the aforementioned separation of "enter trade order" and "validate trade".

Size: Table 4 compares the two process hierarchies in terms of the size metrics introduced previously. Both decomposition methods produce sub-processes that exceed, in terms of activities, the suggested 4–6 or 5–7 activities (however not empirically validated). It is noteworthy that breakpoint-based decomposition resulted in larger number of lazy processes (4 sub-processes with fewer than 4 activities) as compared to the data object-based heuristics (1 sub-process with fewer than 4 activities). If size is evaluated in terms of number of nodes, no sub-process exceeds 50 nodes. The largest sub-process consists of 38 nodes and the average of all sub-processes in each set is well below 50 nodes (cf. Table 4).

Table 4: Size Metrics

Decomposition Heuristics	Min # of Activities	Max # of Activities	Avg # of Activities	Min # of Nodes	Max # of Nodes	Avg # of Nodes
Breakpoint	2	12	6,4	5	23	12,5
Data Object	3	19	8,4	5	38	17,1

More recent research [146] indicates a lower threshold value of 31 nodes rather than 50. Given this threshold value, the breakpoint-based heuristic would be less probable of being error prone. On average, the size of the data object-based set results in 31 % more activities and 36% more nodes per subprocess.

Table 5: Average Size of Process Models

		Lower Quartile	Interquartile Range	Upper Quartile
Activities	Breakpoint	2,5	6,14	10,75
	Data object	3,67	6,60	16,00
Nodes	Breakpoint	5,75	12,29	19,75
	Data object	6,00	12,60	31,67

The distribution of size in terms of average number of activities and nodes across quartiles (cf. Table 5) reveal that data object-based heuristics resulted in similar sizes for the lower and interquartile ranges. However, there is a difference in the upper quartile where data object based heuristics result in larger process models.

Coupling and Cohesion: Previously, A number of coupling metrics that have been proposed in the literature on process decomposition were identified. These metrics do not directly consider the coupling of data objects between sub-processes. However in this case study, the process models contain details of data objects used by the activities of the process. In order to take into account data objects when measuring the coupling, the idea that sub-processes are coupled when they share data objects [81] is followed. Specifically, for this setting, the notion of "coupling between objects" (CBO) metrics introduced by Chidamber and Kemerer [31] is adapted. As such, coupling is measured as the ratio of 'number of sub/processes that share common data objects of other sub-processes' and 'total number of possible connections' (N *(N-1) where N is total number of sub-processes in the set of process models). If a set of process models has a coupling close to 1, then most sub-processes share data objects with other sub-processes. Reversely, low coupling means that most sub-processes use their "own" data objects and are not connected to data objects of other sub-processes.

As such, the definition stating that a sub-process is strongly cohesive if it contains all the activities required to transform an input to an output [81], is adopted. Therefore, cohesion of a sub-process is measured in the same way as coupling but confined to one sub-process rather than between sub-processes. Cohesion is then the ratio between 'number of links between activities and shared data objects within a sub-process' and 'total number of possible shared links (activities * data objects) of a sub-process'. A high cohesion value indicates that most of the activities of a sub-process share data objects and therefore should be modeled together.

The coupling for breakpoint-based set of process models is 10% whereas it is 5% for the data object-based set. As the coupling metric looks at the data objects, this result is as expected. If a data object-based heuristics is applied, the coupling is lower as compared to breakpoint-based heuristics. The cohesion was calculated for each sub-process and an average was calculated for the set of process models. The average cohesion of the two sets is very similar (26% for breakpoint-based and 27% for data object-based). One could expect higher cohesion when using data object-based heuristics but, in this case, the difference is marginal. However, when ordering the sub-processes according to cohesion value (from lowest to highest), and comparing them as shown in Fig. 5, the cohesion of the data object driven sub-processes are consistently higher than those of the breakpoint driven heuristics. Nevertheless, it should be noted that the difference is marginal.

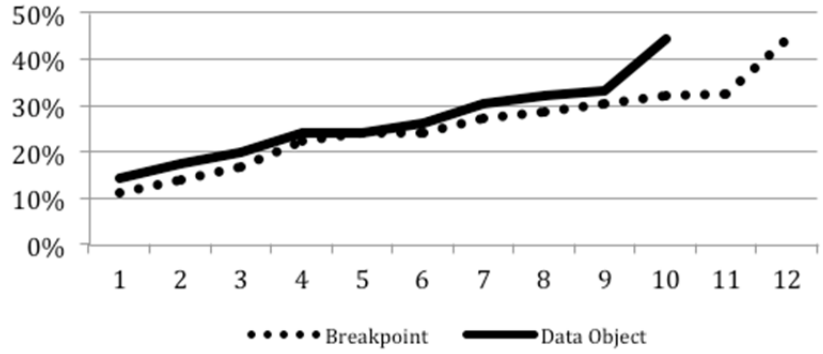


Figure 5: Cohesion for sub-processes using Breakpoint and Data Object Heuristics

Complexity: CNC and density metrics for evaluating the complexity of the two sets of process models were applied. The breakpoint-based set of process models had an average CNC of 1,02 and the data object-based set of process models had 1,0. As to density, the average density is 0,11 and 0,10 respectively. As such, the sub-processes of both sets have a very similar degree of complexity.

Table 6: **Summary of Coupling, Cohesion and Complexity Metric Values**

Decomposition Heuristics	Coupling	Min Cohesion	Max Cohesion	Average Cohesion	Min CNC	Max CNC	Average CNC	Min Density	Max Density	Average Density
Breakpoint	10%	11%	44%	26%	0,83	0,83	1,02	5%	20%	11%
Data Object	5%	14%	44%	27%	1,21	1,21	1,0	3%	20%	10%

3.3.3 Threats to Validity

Case studies come with several inherent threats to validity, particularly regarding external validity and reliability [156]. The main threat to validity is external validity (extent to which results can be generalized). In this case, two heuristics were applied on a structured real-life flat process model and accordingly, the results are limited in the extent they can be generalized. The results could be dependent on the process models being decomposed and therefore, the results should be regarded as indicative rather than conclusive.

Reliability concerns the level of dependency between the results and the researcher i.e. would the same results be produced if another researcher

conducted the study? This threat was tackled by having two of the authors decompose the flat process independently of each other and in different order.

Another threat to validity comes from the use of complexity metrics as proxy for understandability. However, understandability of process models is not restricted do its metrics only. Therefore, exploring the effect of decomposition on cognitive understandability is a relevant and important direction for future work.

3.4 Experiment

3.4.1 Design

In addition to the case study, the effects of two decomposition heuristics are tested in an experiment. In this section, the design of the experiment is described.

The results of the case study showed that there are no significant differences in terms of partitioning of model elements and metrics used as proxy for understandability. Based on these results, the goal of the experiment is to investigate if the different decomposition heuristics (data object and breakpoint) cause any significant differences on size, cohesion, complexity and density metrics. Accordingly, the following hypotheses are formulated.

Hypothesis: Different decomposition heuristics (data object and breakpoint) cause significant differences in size, cohesion, complexity and density metrics.

Alternative hypothesis: Different decomposition heuristics (data object and breakpoint) do not cause significant differences in size, cohesion, complexity and density metrics.

The subjects of the experiment were second year master students of the University of Tartu. The population consisted of 36 (voluntary) students who were in their third month of a course on business process management. As such, they had gained the required background and familiarity needed to read, understand and work with BPMN models. Each student was randomly assigned to one of two groups (decomposition based on data object or breakpoint heuristics).

Both groups received the introduction to the experiment, decomposition of process models, the flat process model and practical instructions such as submission format. The only difference between the introductions between the groups was that they were introduced to the decomposition heuristics they were to use. As such, the group assigned to decompose the flat process model using breakpoint heuristics, was only introduced to the breakpoint heuristics.

The object of the experiment was the same flat process model used in the case study as described above. The students were given 6 days to decompose the flat process model. In addition to the decomposition task, they also answered questions regarding the extent of their previous experience with process models and ratings of difficulty of the experiment. All documents were submitted as image files.

The submitted sets of sub-processes were analysed to ensure that they were complete. Submissions that required interpretation as to which sub-process one or more activities should belong, were discarded. At the end of the filtering, 25 valid submissions remained. Of these, 14 applied data object heuristics and 11 applied breakpoint heuristics for decomposing the flat process model. For each valid submission, number of sub-processes, size as number of activities and number of nodes per sub-process, cohesion, CNC and density for each sub-process was calculated. The average of these values were calculated for the whole set of sub-processes of each submission and used as values for the set of decomposed process models. For instance, if a student had decomposed the flat process model into 10 different sub-processes, the values of each of the ten sub-processes was first calculated. Then, an average of each metric for all the 10 sub-processes was calculated and used in the data analysis presented below.

The two groups were similar in terms of prior experience with process models and familiarity with BPMN. For instance, both groups had on average, created or edited about 8 (7,64 for breakpoint and 7,43 for data object) process models during the past year. The process models created or edited had on average of 18 activities (18,64 for breakpoint and 18,79 for data object). In response to questions regarding familiarity, confidence in understanding and using BPMN, both groups stated that they 'somewhat agree' on average (on a 7-step scale of 'strongly agree', 'agree', 'somewhat agree', 'neutral', somewhat disagree', 'disagree', and 'strongly disagree'). Statistical test conducted to verify their statistical significance (t-test and Mann-Whitney test) showed that the averages of both groups were similar with 95 % confidence.

3.4.2 Results

The decomposed process models were analysed in terms of number of sub-processes, size (as measured by number of activities and nodes), cohesion, CNC and density. For each set of decomposed process model, the average of the above metrics were calculated. The average values of these metrics are shown in Table 7 below.

Table 7: Average Value of Metrics

Heuristics	No of Sub-processes	Size (Activities)	Size (Nodes)	Cohesion	CNC	Density
BreakPoint	16.73	6.48	12.46	0.26	1.01	0.15
DataObject	15.14	5.34	10.08	0.38	0.96	0.15

In order to determine if the average values of both groups are similar (with statistical confidence), the two-sample t-test (for normal distributed samples) or Mann-Whitney test (for non-normal distributed samples) was conducted. In order to determine which of these tests should be applied, the Shapiro-Wilk test (results shown in Table 8) of normality was conducted.

Table 8: Shapiro-Wilk test determining if the data is normally distributed.

Heuristics	No of Sub-processes	Size (Activities)	Size (Nodes)	Cohesion	CNC	Density
BreakPoint	0.990	0.000	0.000	0.953	0.124	0.859
DataObject	0.454	0.001	0.008	0.323	0.004	0.006

The Shapiro-Wilk test shows that the two-sample t-test can be applied on 'number of sub-processes' and 'cohesion' as the p-value for both sets is above 0.05. For the other metrics, which are not normally distributed, Mann-Whitney test is conducted to determine if there are significant differences in the averages of the two sets.

The results show that there is not enough evidence to support the hypothesis that the values of the two sets, in regards to 'average no of sub-processes', 'average size (activities)', 'average size (nodes)', 'average CNC', and 'average density' are significantly different as the p-value is above 0.05 (cf. Table 9). As such, it can be inferred that the average values of the two sets (BreakPoint and DataOjbect) are similar. However, as the p-value for 'average cohesion' is below 0.05 (cf. Table 9), it can be stated, with 95 % confidence, that their averages differ. This is expected as data object heuristics base decomposition on the cohesion of the activities. Any other results would have been surprising.

Table 9: P-value of two-sample t-tests and Mann-Whitney tests.

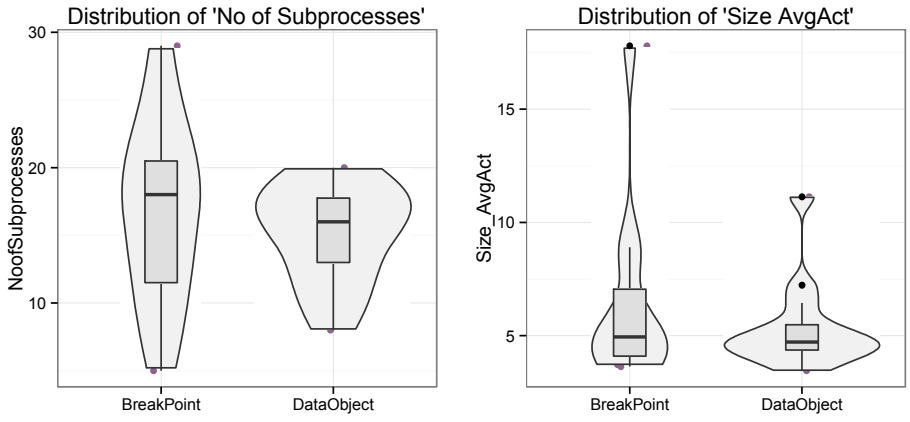
	Test	**P-Value**
Average No of Sub-processes	Two-sample t-test	0.502
Average Size (Activities)	Mann-Whitney test	0.809
Average Size (Nodes)	Mann-Whitney test	0.687
Average Cohesion	Two-sample t-test	0.004
Average CNC	Mann-Whitney test	0.309
Average Density	Mann-Whitney test	0.689

The violin plot and boxplot of the metrics as shown in Fig. 6, visually expresses the same results. One interesting observation can be made from the results. The standard deviation for all metrics in breakpoint heuristics is noticeably higher than for data object heuristics.

Table 10: Standard Deviation

Heuristics	No of Sub-processes	Size (Activities)	Size (Nodes)	Cohesion	CNC	Density
BreakPoint	6.92	4.16	7.63	0.08	0.12	0.06
DataObject	3.65	1.94	3.53	0.10	0.06	0.04

For instance, as can be seen from Table 10 above, the standard deviation of breakpoint set is often about twice as high as for data object heuristics (with the exception of cohesion). Furthermore, the heights of the boxplots in Fig. 6 (encompassing 50 % of the data) for breakpoint heuristics are larger as compared to the data object heuristics. As such, it seems to indicate that by following data object heuristics, more consistency is achieved in regards to number of sub-processes, size, complexity and density. In other words, it seems that breakpoint heuristics allow for more interpretation as to where the logical breakpoints are in the flat process model when decomposing it.



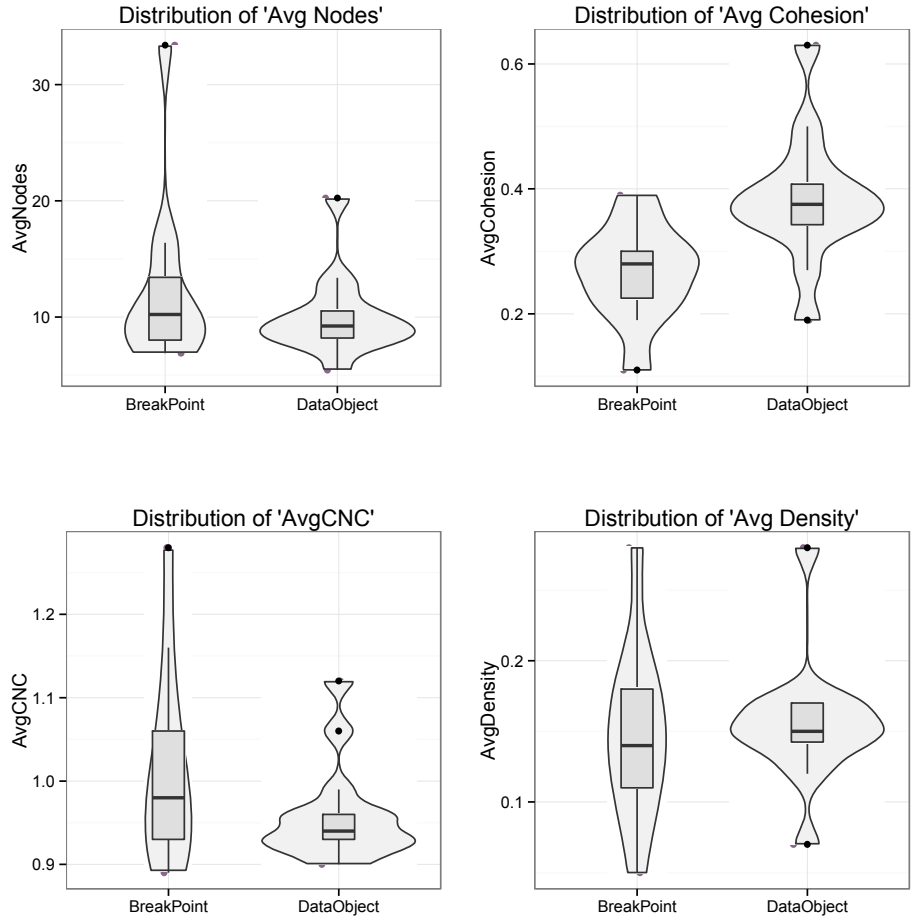


Figure 6: Violin Plot and Boxplot of the Metrics

The questionnaire also included 4 questions regarding degree of difficulties on (Q1) understanding the flat process model, (Q2) difficulty in identifying logical breakpoints or common data objects, (Q3) difficulty in determining which activities should belong to one sub-process and (Q4) difficulty in applying the heuristics when decomposing the flat process model. The responses were given on a scale between 1 and 5 (‘very simple’, ‘simple’, ‘neutral’, ‘rather difficult’ and ‘very difficult’). In order to detect any dissimilarity between the responses given by those who applied breakpoint as compared to data object, Fisher’s Exact test was conducted (cf. Table 11).

Table 11: Fisher's Exact Test

	Q1	Q2	Q3	Q4
Fisher's Exact Test (p-value)	0.5844	0.6006	0.1882	0.3661

A closer look at the results for Q3 and Q4 (Table 12) show that there is a slight overweight of 4 (rather difficult) for both questions from those who got assigned the data object heuristics.

Table 12: Distribution of Responses for Q3 and Q4.

Scale	1	2	3	4	5	Total
Q3 – How easy or hard was it to determine which activities should be in one sub-process model?						
Breakpoint	0.00	0.27	0.55	0.18	0.00	1.00
DataObject	0.00	0.14	0.29	0.57	0.00	1.00
Q4 – How easy or hard was it to apply the breakpoint/data object approach when decomposing the large flat process model?						
Breakpoint	0.00	0.27	0.46	0.27	0.00	1.00
DataObject	0.00	0.29	0.21	0.57	0.50	1.00

The Fisher's Exact test indicate that for Q1 and Q2, the responses given by the two groups are similar (high p-value). However, for Q3 and Q4 (in particular for Q3) there seems to be indications that it was more difficult, when using data object heuristics, to determine which activities should belong to one sub-process (Q3) and applying heuristics when decomposing a flat process model (Q4).

3.5 Summary

The survey showed that three heuristics (repetition, shared processes and role based) are not enough for being applied generally on a set of process models. In other words, applying these heuristics would not produce enough candidates for subprocesses, as sections of process models that do not fulfill the required conditions would not be fragmented to subprocesses. These heuristics are therefore useful as complementary to other heuristics. Three heuristics

(breakpoint, data object and structuredness) provide necessary but not sufficient criteria for decomposition. These heuristics can be applied on processes when decomposing but fall short in determining which process fragments to model as a subprocess. It is possible to set threshold values that determine the fragments to include in a subprocess such as in the case of structuredness (metrics such as size). It should be noted that there are no evidences or indications on what treshold values should be put for different metrics. Even with arbirtarily chosen values, such methods are not sufficiently refined yet and fall short when compared to human approaches [138]. As such, there are currently no heuristic that provide both necessary and sufficient criteria for decomposition.

In this case study, the breakpoint and data object-based heuristics were applied on a real-life flat process model. The case study showed that applying different heuristics results in different structures. Using quantitative metrics as approximation of understandability, the case study showed that the two heuristics are similar in terms of understandability.

The experiment confirmed the results of the case study. However, the experiment indicated that it might be more difficult to apply the data object heuristics. Furthermore, it seems that breakpoint heuristics allows for more loose interpretation as to where breakpoints are as compared to data objects. This might suggest the need for more guidance as how to find logical breakpoints. On the other hand, applying data object heuristics presuppose that all activities are connected to at least one data object. If the process model includes activities that use or produce data objects that are not captured in the process model, or has activities that simply do not use or produce data objects, it is not clear to which sub-process that activity should belong. As such, data object heuristics will not be able to offer clarity.

4 FOUNDATIONS OF PROCESS VARIATION

This chapter discusses the foundations of process variation by introducing business processes and their variations in Section 4.1. Following this, Section 4.2 discusses where variants occur in a process model followed by definition of what constituted as a viable variant in Section 4.3. Section 4.4 presents a meta-model of process variation and Section 4.5 validates the concepts of variations and viable variants. Then, in Section 4.6 and 4.7 business and syntactic drivers of variations are discussed respectively. Finally, this chapter is concluded by a summary in Section 4.8.

4.1 Business Process

Business processes have been the object of study for a while with its origins dating back as far back as to the 1950-ties [25]. At such early stage of research, any sequence of work activities was considered to be a business process. However, during the beginning of the 1990-ties, a renewed interest was born for business processes when American corporations sought to re-engineer their processes in order to gain significant competitive advantages. That also resulted in more proper attempts to define business processes. Davenport [36] defined a business process as "*a structured, measured set of activities designed to produce a specific output for a particular customer or market. ... A process is thus a specific ordering of work activities across time and space, with a beginning and an end, and clearly defined inputs and outputs: a structure for action. ... Processes are the structure by which an organization does what is necessary to produce value for its customers.* Hammer and Champy introduced the transformational aspect of a business process when defining it as "*a collection of activities that takes one or more kinds of input and creates an output that is of value to the customer*". These definitions are more business oriented but the same elements can be found in more general definitions (including the IT perspective), such as "*a collection of inter-related events, activities and decision points that involve a number of actors and objects, and that collectively lead to an outcome that is of value to at least one customer*" [54] or "*a set of activities that are performed in coordination in an organizational and technical environment. These activities jointly realize a business goal*" [190]. In summary, all definitions include the elements of coordinated performance of a set of activities that uses inputs and aim at fulfilling a certain predefined outcome or business goal for a receiver such as a customer.

A *business process model* is a representation of a particular business process that expresses the relationships and restrictions of the activities of the process, using a set of notational techniques [190]. Within a collection of process models, it is not uncommon to find variations. For instance, as mentioned before, an insurance company would typically perform the process for managing claims differently depending on whether it concerns a personal, vehicle or property

claim [52]. Each of these different processes relating to managing claims, are *variants* of the process [71].

4.2 Variation Points and Decision Points

Given a process model or a collection of process models capturing a family of process variants, there will necessarily be points at which a choice is made between multiple branches. For instance, when using the Business Process Model and Notation (BPMN), such choices may appear in the form of exclusive gateways (a.k.a. XOR-splits), inclusive gateways (a.k.a. OR-splits) or other types of split gateways. Such points are hereby called explicit *branching points*. Another type of branching point (implicit) occurs when a choice is made between instantiating one versus another process model, such as instantiating a process model for handling personal claims versus instantiating a process model for handling vehicle claims.

Each branching point corresponds either to a potential *variation point* or a *decision point*. The term variation point is very familiar within the field of Software Product Line Engineering (SPLE) and Feature Diagrams (FD). Weiss and Lai [189] define variation in product family as "*an assumption about how members of a family may differ from each other*". Based on this assumption within the domain of engineering, a variation point is a decision (branching) point together with its outgoing options. Each available option branching out from a variation point (functions or qualities in the context of SPLE), is then defined as a variant [73]. In other words, within the domain of FD, there is no relevant and significant distinction between a variation and a decision point. As such, a variation point within this field depicts or states the differences from a static point of view.

However, in a business process management setting and in particular, the enactment of a process, there is a need for distinguishing a variation point from a decision point. The distinction between a variation point and a decision point is often referred to as build-time versus and run-time [153], or as design-time and run-time [6, 69]. In the context of this thesis, the distinction given in the definitions above [6, 69, 153] is chosen. The distinction is that if a variant stemming out of a branching point, has to be executed (given the criteria are fulfilled) in order for the process to be successful [30], it is identified as a potential variation point. However, if the choice at the branching point is dependent on the circumstances at run-time [30, 159], it is defined as a decision point.

A *variation driver* is a parameter or criterion that is used at a variation point to distinguish between its branches. A *variation option* is a possible option that exists at a variation point. Concretely, a variation option is a value or range of values of the variation driver associated to a variation point.

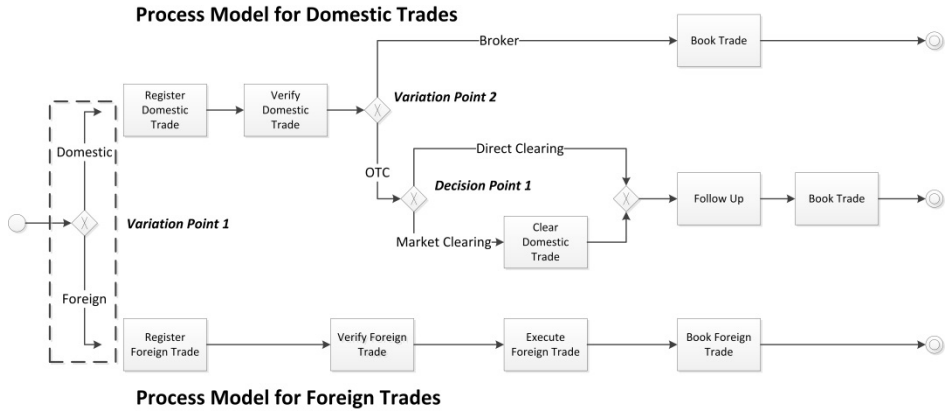


Figure 7: Illustration of Definitions

Fig. 7 depicts two models for the equity trading processes (domestic or foreign equity). The process models for domestic and foreign equity capture two variants of the equity trading process. The process model for domestic equity covers two variants: one for trades via a broker and another for trades made over-the-counter (OTC).

Seen collectively, these two process models contain three branching points. The first branching point (variation point 1) is the one where a choice is made between instantiating the "domestic equity" process model or the "foreign equity" model. This branching point is an example of an implicit *variation point* as its branches lead to different but similar outcomes (trading an equity). In Fig. 7, this variation point is represented inside a dotted rectangle. Importantly, the XOR-split gateway inside this rectangle does not exist in the process models. It is added to the figure for the sake of making the branching point visible. In reality, the branching point exists merely by virtue of a choice being made between instantiating two alternative process models.

Within the process model for domestic equity trading, there is an *explicit variation point* (variation point 2) where a choice is made between using a broker or trade OTC. Within the process model for clearing a domestic equity, a third branching point can be found with two branches ("direct" vs. "market clearing"). This is a *decision point* as it is a decision taken based on the context of the run-time execution of the process.

The *variation driver* at variation point 1 is "domestic vs. foreign", and at variation point 2 it is "broker vs. OTC". At variation point 1, "domestic" is a *variation option* and "foreign" is the second variation option. Similarly, at variation point 2, broker is the first variation option and OTC is the second.

4.3 Viable Variant

Within the domain of goal modeling, a goal is the objective that a certain system is to achieve [99] and extending this definition to business processes, an output is the business goal that the activities of a process realize [190]. Inputs are the preconditions needed for a certain business process, commonly represented as a start event in business process models (design time). A viable variant will start at an *implicit* or *explicit* variation point. Furthermore, two process models can only be variants of each other if they belong to the same domain or are somehow, from a business perspective, related to each other. As such, they should belong to the same "meta-process" [168].

A viable variant is therefore defined as the outgoing path from a design-time variation point, when the different outgoing paths have similar inputs and/or lead to similar outputs as perceived by a domain expert. If a variation point results in two or more different variants, each leading to an output that a domain expert considers belonging to clearly different business processes; it is not considered to be a viable variant. The definition of a viable variant is therefore contextual.

4.4 Meta-Model

The meta-model of this framework as shown in Fig. 8, gives an overview of the presented definitions. The top-level concept in this meta-model is that of a process. A process is a collection of logically related activities. It should not to be confused with a process instance, which is one specific execution of a process, nor should it be confused with a process model, which is a specific way of describing a process or part of a process.

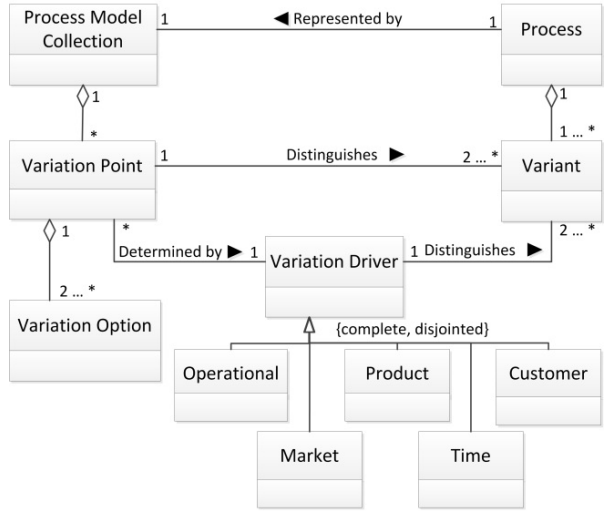


Figure 8: Meta-Model of Variability

A given process is represented by a collection of process models. Within a collection of process models, there are variation points, each of which will have at least two variation options. The variation point has one variation driver. It may happen that one can identify multiple variation drivers in a variation point but if so, these variation points could be split into two or more consecutive variation points so that each of them will have only one variation driver. Therefore, the assumption is made that each variation point has a single variation driver. It should be noted that in the meta-model, the concept of variation drivers has several sub-classes (complete and disjointed) that will be elaborated upon later in this chapter.

4.5 Validation

As a preliminary validation, the definition of viable variants was applied on two collections of process models. The first collection of process models is from a full-service (retail and commercial) bank, operating mainly in selected European markets. The second collection is from a governmental agency providing an array of services related to land management (including maps and satellite images). By analyzing the processes related to the back-office processing of equities in the banking case, and by analyzing processes related to managing document processing in the second case, the applicability of the definition of viable variants is tested. In other words, the question being examined is: (1) can the definition of viable variants be applied for identifying variation points from a collection of process models?

4.5.1 Background

The first case is from a mid-sized European bank that covers the entire spectrum of banking products such as retail banking, life insurance, and investment banking with more than 700 branches. This case covers the processes involved in equity trading services in one of its subsidiaries. The collection of processes covers the back-office operations of domestic and foreign equity trading. The collection of processes consists of 8 top-level processes that are considered to be variants of one another. Each of the 8 top-level process models can be decomposed into sub processes, leading to a total of 30 process and sub-process models.

The second collection of process models is from a governmental agency dealing with various issues related to land ownership and survey information. This case concerns management of documentation processing. There are 9 top-level process models and additional 15 sub process models. In total, this collection is comprised of 24 process models. These process models cover the business processes of two geographical areas (differences of these two geographical areas have been captured in the process models using annotations).

These two cases, together, consist of 17 top-level process models and 37 sub process models making it a total of 54 process models.

4.5.2 Identifying Viable Variants

In analyzing the data, the first step was to identify the variation points in order to identify all the variants in the consolidated processes. Using the definition of variation points, all branching points were analyzed and designated either as a variation point or a decision point. This was achieved by identifying each variation point by assessing if the outgoing branches of that point belonged to different but similar outcomes. If the different paths stemming out from a candidate of variation point are considered to belong to the same variant, it was classified as a decision point.

For illustration (Fig. 9), lets consider a sub-process model for calculation of fees. The first step is to identify all XOR splits in the process model. The first one occurs just after the sub-process called "Get Product Details". As the outgoing branches can be considered to be variants (both leading to similar outcome but in different ways), it is defined as a variation point. The variation driver is "Counter or Online Customer" and the variation options are identified as "Counter" and "Online". That is, at this variation point, the next step of the process model is dependent on it being counter or online customer. The second XOR split is defined as a decision point. At this point, the question of "priority or not", determines the next step in the process model. However, both alternatives are within the same variant, as they lead to the same outcome. Therefore this point is classified as a decision point.

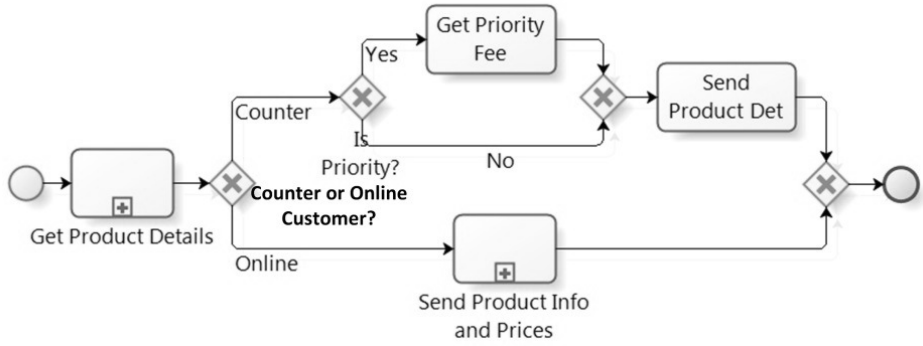


Figure 9: Example of eliciting variation point and driver in a process model

4.5.3 Findings

The implicit variation driver in the collection of processes for equities was along product, which was domestic versus foreign equity. No additional variants were identified from the collection of 8 top-level process models. However, the analysis identified an additional 6 implicit viable variants in the process models making it a total of 7 variable variants. The additional implicit viable variants identified, can be labeled as Counterpart type and Execution type (Table 13).

Table 13: Analysis of Variation Drivers in the Bank Case

Equity Type	Counterpart Type			Execution Type		
Domestic vs. Foreign	Own vs. Custody	Own vs. No Custody	Custody vs. Without	Exchange vs. OTC	Exchange vs. Broker	OTC vs. Broker
2	3	1	1	2	1	1
2	4			4		

By counterpart type is meant a variant determined by what kind of counterpart or customers the trades are being made against. The types identified are “Own” (when the bank is making a trade for itself), “Custody” (when the bank is making a trade on behalf of a client who has a custody service agreement) and “Without” (when the bank is making a trade for a client who does not have a custody service agreement). Execution type refers to how the trade is made. It could be “Exchange” (when the trade is made over the regulated domestic exchange stock market), “OTC” (when the trade is made as a bi-lateral agreement between two parties outside the exchange) or via a “Broker” (when an intermediary is used to make a trade).

It is noteworthy that the set of input process models were organized according to variants that had the fewest occurrences (domestic vs. foreign). Counting, the analysis showed that equity type was responsible for 2 occurrences of variations, whereas counterpart and execution type caused 4 variation points. This indicates that the definition of viable variants could be used for quantifying to what extent each variation driver is responsible for variants in a given collection of processes.

In the second case, 8 additional variation points were identified. Of the additional identified variants, 3 are related to product and two are related to customer. Within product type, the analysis revealed 3 distinct variants. The first one concerned type of transaction (NASF vs. non-NASF), the second was related to number of transactions (single vs. multiple package), and the third re-

ferring to what kind of property deed is being processed. As to customer type, the first variation is related to how the customer has come in contact with the agency (online vs. over the counter) and the second refers to new versus existing customers.

Table 14: Analysis of Variation Drivers in the Governmental Agency Case

Area	Type of Product			Type of Customer	
South vs. North	NASF vs. non-NASF	Single vs. Multiple Package	Type of Deed	Type of Contact	Type of Customer
9	2	1	1	2	1
9	4			3	

In addition, a candidate variation point related to managing refund of payments was identified, but not defined as a variation point. This is because it could be considered to be variants within the sub-process of payments and not of the overall process of management of documentation processing. However it could be defined as a variation point depending on the objective of the analyst and on what granularity level the analyst is working with, as discussed in previously.

In summary, the analysis of two collections of process models consisting of a total of 54 process models indicates that the definition of viable variants can be applied for identifying variation points.

4.6 Business Drivers of Variations

Variations of business processes exist because of a reason. There can be many reasons why an organization chooses to design or maintain variants of a process. The root causes can range from externally dictated reasons such as national laws, specific internal choices such as cultural reasons or a mixture of them both [122]. By ordering the many root causes of process variations into classes of variation drivers, a reduction of complexity is achieved [17]. This enables working with a few classes of drivers that one has some information about, rather than with a multitude of unique root causes [171]. For this purpose, there is a need for "special classification" where focus is on one attribute of interest (root cause of variations) as opposed to a "general classification" that attempts to group objects based on many attributes [111]. The developed framework here is a product of deductive research and therefore a typology (and not a taxonomy) as defined by Bailey [17]. A typology has the limitation of being the product of a researchers understanding and may therefore not necessarily reflect reality. Typologies are therefore simple classifications aimed at being used for

specific purposes [98]. However, the great advantage of a typology is its ability to reduce complexity for specific purposes such as classifying root causes into few classes of variation drivers [17].

4.6.1 Classification of Business Drivers

The theoretical base of this framework is built on the business architecture layer of enterprise architecture presented by Rummler and Brache [155]. In their framework, organizations are viewed as systems, which operate within a larger "super-system". This super system is the context within which an organization operates and the reality to which it must adopt itself to in order to survive. According to this framework, the environment, resources, stakeholders, markets, customers and competitors influence organizations. Operating under the influence of these external variables, all organizations create an output by procuring resources in order to manufacture a product or produce a service. These products and/or services are then brought to a market place where the customers, those who need or wish to consume the manufactured outputs, can buy the products and/or services.

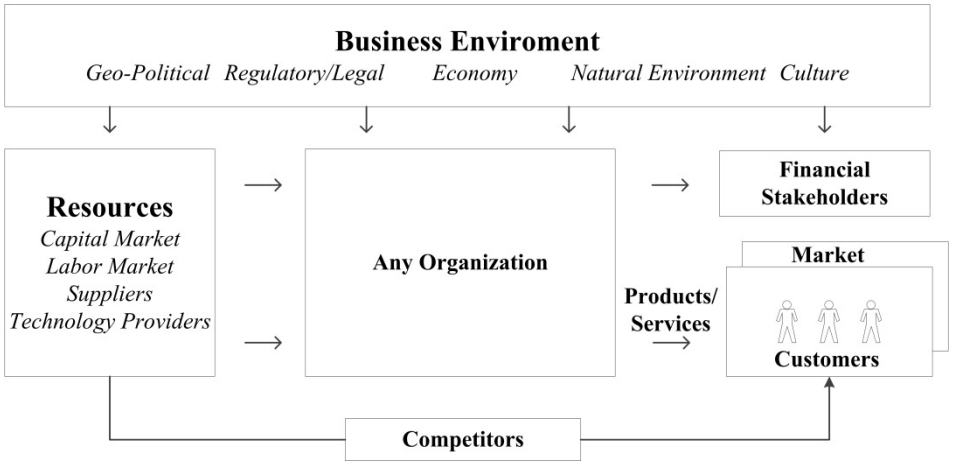


Figure 10: Rummler and Brache framework adapted from [62]

Rummler and Brache framework can also been viewed as a map of factors that an organization needs to relate to in conducting its business. An organization interprets its business environment and chooses to respond to it in ways that they perceive will ensure competitive advantage. Therefore, these factors have an impact on the business processes of organizations. As such, these factors, combined with how an organization decides to respond and manage them, are causes of variations in business processes. The premise of this framework is that

these decisions will manifest themselves in business processes as variation points.

Rummler and Brache framework, on its own, does not include an explicit classification of variation drivers occurring in business processes. But by overlaying the W-questions (how, what, where, who and when) on Rummler and Brache framework (Fig. 11), a system for assessing and classifying variation drivers is obtained.

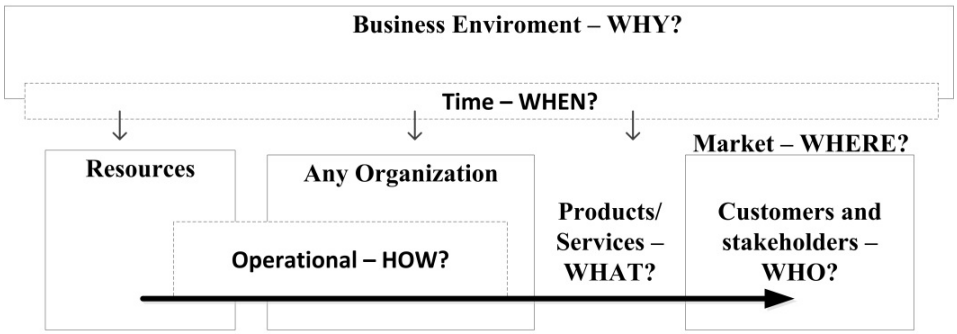


Figure 11: A framework for business variation drivers

Behind each variation, there is an organizational specific reason that corresponds to the w-question "why". All these different root causes can be classified as follows. Organizations have a set of processes to procure resources and manufacture (*how*) output (*what*) that they bring to a marketplace (*where*) for customers (*who*) to buy. Finally, organizations sometimes (*when*) adapt their processes to a specific external situation in order to remain efficient throughout the value chain.

Rummler and Brache framework include "competitors" as a factor, but it is excluded it in this analysis since it affects the processes indirectly. Competition drives companies to increase their performances (such as to cut cost, improve quality or shorten product development cycles) in order to gain or maintain a competitive edge. As such, competitors affect an organization indirectly and organizations adapt or respond by improving their business processes for better value production. As such, the influence of competitors is expressed through the five categories already elicited.

In summary, the key tenet of the adapted framework presented here, is that business drivers for variations in business processes, based on their causes, can be classified as *operational (how), product (what), market (where)*, *customer (who)* and *time (when)* drivers. This classification and their corresponding questions can be used, as will be shown later, to systematically elicit business variation drivers.

4.6.2 Classes of drivers

The above analysis leads us to recognize five orthogonal categories of variation drivers, namely: operational (*how*), product/service (*what*), market (*where*), customer (*who*) and time (*when*).

4.6.2.1 Operational Variations

Every organization has designed processes to manufacture what will bring value to its customers. Although traditionally manufacturing processes has been referring to the production of physical products, manufacturing process is considered to covering services as well in accordance with the broader definition proposed by Kakaomerlioglu & Carlsson [84].

The processes of Dutch municipalities has been investigated by Buijs *et al.* [26] who compared the processes for building permit and housing tax in four different municipalities. Gottschalk *et al.* [186], using the same data set, compared the process of acknowledge an unborn child. Buijs *et al.* chose those municipalities that had the same type of information system and yet, each of them had different processes for building permit and housing tax. Gottschalk *et al.* chose municipalities that varied from each other in regards to information systems used. In these cases [26, 186] the municipalities are offering the same service but have chosen to manufacture them differently. These variations exist as the municipalities have a certain degree of autonomy and are free to choose how to design these processes and what system solutions to use. The variations in this example are manufacturing driven variations as in choosing between two variants, the answer to the w-question "*how*" provides guidance as to which variant to follow.

4.6.2.2 Product/Service Variations

The primary purpose of any given organization is to produce value in the form of products and/or services. As firms offering multiple products/services are ubiquitous, the field of multi-product competition and product differentiation strategies have been and is being studied extensively [106]. Offering several different products or a set of products with differing features is therefore a driver of variations in business process models.

La Rosa *et al.* [152] presents an example from the film industry. In this example, there are two variants of the post-production process of a film. The first variant is for when shooting the film "on tape" and the second for when the film is shot "on film". These two variants follow the same path until a certain point where the variation occurs. When the case of "on tape" is relevant, there occurs "online editing", and when the film is "on film", "negmatching" takes place. This variation point is driven by product/service as the product, in this case "*what"* kind of film (tape or film), determines which path the next step will

follow. Another example presented by La Rosa [151] refers to an insurance company and its processes for handling claims. In this example, an insurance company has similar processes for handling claims related to motor insurances and personal insurances. The process follows the same path but diverges at a certain point. The driver for the variation, that is, which variant the process follow next, is determined by *"what"* kind of insurance is dealt with (motor or personal). Van der Aalst *et al.* [5] uses an example of travel requisition. This process covers two variants, one for international and one for domestic travel. If it concerns an international travel, the process involves requesting quote, preparing travel requisition form, submitting for approval, approval or rejection of the request, possible modifications or updates of the request, and re-submission or cancellation. For domestic travels, the process includes asking for quote and reporting the request to the administration. This variation is driven by product/service as the question *"what"* kind of travel suggest which of the two variants is relevant.

4.6.2.3 Market Variations

The concept of segmenting the market that an organization targets with its products/services (market segmentation) has been studied extensively since it was introduced in the late 1950s [188]. Market segmentation can be defined [42] as dividing a heterogeneous market into relative homogenous segments. Organizations can and do segment their markets differently in accordance with their own needs and preferences [61]. The many different methods and the various basis for market segmentation studied [188], illustrates the variety of organizational flexibility in market segmentation strategy implementation. As organizations can divide their markets into different segments and approach them differently, their business process models will have market driven variations. In these variation points, the w-question that is most relevant is "where".

Hallerbach et al. [72] describes the variations of a process for vehicle repair. One of the variations in this process depends on the country. If it is in country 1, the process is described as reception, diagnosis, repair and hand over. The same process in country 2, has a "final check" before the vehicle is handed over to the owner. This variation, as explained in the article, is due to a legal requirement in country 2 stating that the vehicle must be checked before handed over to the owner. This regulation does not exist in country 1 and therefore there is a variation. This is a market driven variation because at the variation point, the answer to the w-question "where" provides the answer as to which variant is relevant. Ebay [122] has processes related to customer management which differ from each other depending on which site it concerns. The processes for customer management in North America, Germany and United Kingdom all differ from each other in regards to phone system, CRM system and the workflow. These variations are market driven as the question "where" gives the best answer as to which variant to pursue.

4.6.2.4 Customer Variations

Organizations produce products/services that bring value to customers but not all customers are the same. Customers can therefore be segmented, that is, divided into various subgroups based on certain attributes and characteristics, and subsequently treated or managed differently [177]. An example of customer segmentation taken from the airline industry is that first class customers are treated differently (have different processes) compared to economy class customers [173]. Organizations therefore, have different processes in offering the same product to different types of customers. Due to this, customer is a driver of variation in business processes.

Van der Aalst [3] presents Salesforce, a company that offers solutions for the sales processes of other companies. This solution gives the companies access to a shared infrastructure while allowing for customer specific configuration (as each customer has its own specific needs, requirements and preferences) causing variations in the Salesforce processes. The customer therefore, drives the variation (*"who"*). Kleinj & Dekker [90] writes about the inventory of "rotables" (aircraft part that can be repaired if broken) where a major airline has founded a company to service them with the inventory of such aircraft parts. This company also provides other customers (airlines) with the same service. There is however variation in the process depending on *"who"* the company is dealing with. If it is the major airline that founded the company, there is an agreement that parts are to be supplied within 24 hours in 95% of the times. Similar, but not identical procedure, exist with other airlines that has an agreement with the company. Airlines that do not have an agreement can also approach the company for service. In such cases, the company can decide, depending on which alternative provides the best return, to sell, loan or exchange the part. The variations in this process are caused by *"who"* the customer is and therefore it is a customer driven variation.

4.6.2.5 Time Variations

The above presented variation drivers share the commonality of being independent of differing requirements that may occur in the environment of the process. These process variations do not include the possibility of different execution paths depending on extrinsic events or requirements. If, at a variation point, the path of execution is determined by an external factor, it is defined as a time driven variation. At such variation points, the relevant w-question to determine the next step in the path of execution is "*when*".

An Australian insurance company [8] has call-centers to manage incoming claim calls that are then routed to the back-office that manages the claims. The call-centers have an even flow of calls coming except for during the Australian storm season. During the storm season, the number of calls increases from average 9 000 to as much as 20 000 calls per week causing significant burden on not only the call-centers but also on the back-office who has to evaluate and man-

age the claims. In order to manage this increased burden, the insurance company has created an "event-based response system" [8], based on the severity of the storms. For each category of storm severity, there is a specific process. There are therefore variations in the process depending on if it is storm season and how severe the storm is (four categories). The variant to be executed is dependent on "*when*" (storm season or not) and also on "*when*" the storm is of what category, thus making it a time driven variation.

4.7 Syntactic Drivers of Variations

Two or more variants will most likely have syntactic differences, i.e. they produce their outcomes in different ways. The degree of similarity or differences in how two or more variants execute their processes is relevant for how these are modeled. In general, the more differences there are in how two or more variants produce their outcomes, the more convenient will it be to model these variants separately rather than together. This driver for modeling two variants together or separate, is referred to as *syntactic drivers*, i.e. the decision is driven by the degree of similarity or difference of the variants.

Although there are different notational languages for modeling business processes (such as BPMN [192], UML AD [193] or C-EPC [153]), most of them capture the same set of perspectives. These are functional, behavioral, organizational and informational perspectives [103, 135]. The functional perspective is represented with the process elements *activities* or *tasks*. Activities are either *atomic* (elementary business function) or *complex* (aggregation of a set of related activities and usually modeled as a sub-process) [103, 135]. The behavioral perspective captures the execution sequence of the process elements with for instance flow objects such as *gateways* [12, 103, 135]. The organizational perspective will show who, such as *roles* and *organizational units*, is performing an activity. Finally, the informational perspective captures the data objects that are used and produced in a process [103, 135].

Syntactic variability occurs when two or more variants are (1) executed in two or more visibly different ways (functional and behavioral perspectives), (2) differences in the data objects (informational perspective) used as input or produced as output [134, 165], and/or resources (organizational perspective) performing the activities [110, 149].

4.7.1 Activity Based Similarities of Variants

Sematic similarity compares the similarity of two labels of for instance a task or activity using string-edit distance [43, 51]. String-edit distance measures the number of "atomic string operations" needed (such as deleting, substituting or inserting a character) in order to get from one label to the other label (being compared with) [43]. The fewer operations needed to achieve this, the more similar the labels are.

Contextual similarity extends the similarity comparison of two elements to include the elements that precede (the input context) and succeed (the output context) the compared elements [43, 51]. As such, the context in which the elements exist within is taken into consideration.

Structural similarities move to one level higher above the elements and measure the similarities of the structures of the process models. The basis of these similarity comparisons is that process models are in essence graphs and therefore, graph-matching methods can be used to assess the similarity of two process models [43, 53]. Graph-edit distance is similar to string-edit distance but instead of atomic string operations on labels, graph-edit operations apply the same logic on the graphs (deleting, inserting or substituting nodes and deleting or inserting edges). Similarly, the fewer graph-edit operations it is required to match one process model to another, the more similar they are.

Behavioral similarity, on the other hand, compares the execution semantics of two process models where each complete execution is taken as a trace and used to compare against the set of completed traces of another process model [43, 44, 53].

4.7.2 Data Object Similarities of Variants

Another aspect of similarity between two process models concerns the data objects used as input or produced as output of the activities of the process models. For instance, an activity (task) of a process model might use different data object inputs and depending on which input it uses, have different procedures (for that activity). As such, there is variability due to the data object used as input, or in other words, a data object driven variability.

A data object can be tangible (physical objects such as products and documents) or intangible (such as data fields and logical concepts such as product types) [163]. If objects share the same or similar properties, they can be classified as belonging to the same object type. For instance, consider an order. A business process might have several different templates for orders, one being purchase order, another being customer order. The purchase order is an object and the customer order is another object. Each customer order will have its own order id, date, order items. All customer orders will have the same or similar set of properties. As such, all customer orders can be classified as object type "customer orders" and be represented by its attributes [190]. *Data object variability* is therefore defined as variability caused by using or producing different data objects in a sub-process or activity. Variability in an activity occurs when the activity has a procedure that is not modeled but is different depending on data object input or output. For instance, the activity "register new client" might require different "forms" depending on type of customer. For each form, there might be a different procedure for the same activity, namely "register new client".

4.7.3 Resource Similarities of Variants

A resource in a business process is an participating entity (human or non-human) that performs the work (activity) [54, 190]. Within a process, there can be participants of different types such as separate organizational units (such as customer), different business units within one organizational unit (such as sales or public relations), different sub-divisions of a business unit (such as back-office support for corporate and private clients), and different roles within the same sub-division of a business unit (such as insurance agent managing simple claims and another managing advanced claims) [54, 135, 155, 190]. If two different resources perform a sub-process or activity, there is a variability induced by the resources involved. *Resource variability* is therefore defined as variability due to different resources involved in performing the same sub-process or activity.

4.8 Summary

This chapter on foundations of process variants has covered the concept of variations in process models, their drivers, and a definition of a variable variant as "*the outgoing path from a design-time variation point, when the different outgoing paths have similar inputs and/or lead to similar outputs as perceived by a domain expert"*. The chapter also examined different variation drivers, namely, business, syntactic, data object, and resource drivers. The occurrence of variations, when captured in process models, can impair understandability and maintainability of the process models. In order to reduce the complexity of process models (caused by many variants), they are decomposed along two different lines, vertical and horizontal. Vertical decomposition is mainly concerned with representing a sub-process at a higher level of detail. However, such decomposition does not consider the variants of the sub-process. By applying incremental decomposition on a family of process variants, the problem of determining whether a given process should be modeled in a fragmented or consolidated manner, is reduced to that of deciding whether each of its sub-processes should be modeled in a fragmented or consolidated manner. To guide this decision, a decision framework, based on two classes of variation drivers, is proposed. On the one hand, there may be business reasons (business drivers) for two or more variants to be treated as separate processes (or as a single one) and ergo to model these variants separately (or together). On the other hand, there may be differences in the way two or more variants produce their outcomes (syntactic driver), which make it more convenient to model these variants separately rather than together or conversely. As such, the sub-process can be seen as being "sliced" into its different viable variants (cf. Fig. 12).

The business drivers, as introduced can be classified as *operational (how), product (what), market (where)*, *customer (who)* and *time (when)* drivers. This classification and the corresponding questions can be used to systematically elicit business variation drivers.

The second family of drivers determining whether two variants should be modeled together or separately is syntactic drivers. If each variant were modeled separately, differences in the way variants produce outcomes would naturally be reflected as differences between the models of each process variant. If these models differ in significant ways, it is more convenient to keep them separate, as consolidating them increases complexity and reduces comprehensibility. Indeed, La Rosa *et al* [150] show empirically that the complexity of a consolidated model of two variants (measured by means of well-known complexity metrics such as size, density, structuredness and sequentiality) is inversely proportional to the similarity between the corresponding fragmented models, where similarity is measured by means of graph-edit distance. Hence, if we had a separate model of each variant, we could assess the complexity trade-off between merging them and keeping them separate, based on their graph-edit distance. However, this requires that (*i*) the models of the separate variants are available; and (*ii*) that they have been modeled using the same notation, at the same level of granularity and using the same modeling conventions and vocabulary. These assumptions are unrealistic in many practical scenarios where models of each variant might not be available to start with, and even if they were available, they would typically have been modeled by different teams and using different conventions.

When these assumptions do not hold, the proposition of this thesis is to assess the similarity between variants of a process (or sub-process) by means of subjective judgment of the expected differences between the separate models of these variants. In other words, given two variants, domain experts are asked the question: How similar or different are the separate the variants (models)?

These two concepts are used in determining how to slice a sub-process. As the sub-processes are decomposed (vertically) further and the level of detail of the process models increase (capturing data objects and resources), decisions on modeling a sub-process or activity, is taken based on their induced variability due to data objects and/or resource drivers.

The horizontal decomposition can be seen as "dicing" the sub-process (cf. Fig. 12).

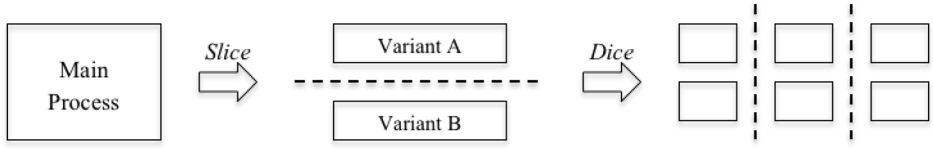


Figure 12: Slicing and Dicing of a sub-process

The survey presented previously, showed that two heuristics have necessary and sufficient criteria for being complete (breakpoint and data object-based). The other heuristics, while valuable and useful as complimentary to each other, are

not generally applicable as independent heuristics for decomposing a process model. The results of the comparison of these two decomposition heuristics showed that applying different heuristics does indeed result in different structures but, based on size, cohesion and complexity of the sub-processes, they are rather similar. As these metrics are approximations for understandability of process models, it is observed that different heuristics do not affect the understandability of process models.

In the following section of this thesis, these conceptual foundations and findings are operationalized in the form of a method for consolidated modeling of families of process variants.

5 DECOMPOSITION-DRIVEN METHOD

It has been shown that current approaches for managing families of process variants when modeling them, are mainly based on process model similarities. As such, the root cause of variability is not taken into consideration when modeling the families of process variants. In this chapter, a method that considers both the business reason and the degree of similarity of variants when modeling families of process variants is presented.

This chapter is structured as follows. Section 5.1 discusses the contexts in which the decomposition driven method can be applied, followed by an overview of the method itself in Section 5.2. The main part of this chapter consists of Sections 5.3 to 5.10, which presents each step of the method in detail.

5.1 Discovery, Consolidation and Standardization

Families of process variants need to be managed when discovering process models or consolidating process models. The method described, can be used both for when consolidating existing process models and for process discovery, i.e. when no prior process models exist. There are slight differences in the implementation of the method depending on if it's a matter of consolidation or discovery of process models. These differences are discussed together with the description.

The method can also be applied for business process standardization purposes. Such projects require efforts to identify candidate business processes that are to be standardized. As such, business process standardization will most likely, at some point, include either discovery or consolidation of business processes. For instance, if a company wishes to standardize a set of their business processes, they need to discover the current processes (as-is models) in order to understand what and how to standardize. With the as-is models, the company might wish to design new process models based on consolidating (with modifying) their process models. However, before processes can be standardized, it is necessary to identify which processes to standardize. With the process models discovered, factors such as complexity of process models under consideration for standardization can be assessed. More importantly, the method can be used to identify viable variants, that is, identifying processes that are variants and therefore, from a business perspective, meaningful to consider for standardization. The application of the method is, nevertheless the same as for discovery or consolidation.

5.2 Overview of the Method

The proposed method for consolidated modeling of process variants follows the idea that decisions on whether to model two or more process variants together or separately, should not necessarily be taken at the level of the top-level pro-

cess (main-process). Instead, such decisions should be postponed to each level of decomposition at the level of sub-processes. In other words, decisions on whether to model in a consolidated or fragmented manner should be interleaved with process decomposition steps. For instance, if two variants are extremely different, it is not optimal to start modeling them together as the number of branching points will limit the readability of the model and most likely result in modeling the variants as separate models. Conversely, if two similar variants are modeled separately, the amount of duplication will be very high. As such, there will be an optimal point in the process hierarchy where decisions to model sub-process together or separately are to be taken. Depending on the particular process, that decision might happen at a higher or at a lower level of process hierarchy. Moreover, as the process is decomposed into sub-processes, a consolidated modeling approach for each sub-process should be the default option until it becomes clear that a fragmented approach is preferable from a business or syntactic perspective.

These ideas are embodied in the following seven steps as summarized below.

1. Model the main process – the purpose of this step is to elicit the main steps (sub-processes) of the business process in question.
2. Identify variation (business) drivers – in this step, the business drivers of variations are elicited so it becomes clear what drives the business process to have variability in the way it produces its outcome.
3. Assess the relative strength of the variation drivers – in this step, the business drivers are analyzed to determine which driver is the most important (strongest) driver of variations in the business process.
4. Identify the variants of each sub-process – at this point the actual existing variants of each sub-process previously elicited (in step 1) are identified and listed.
5. Perform similarity assessment of variants of each sub-process – at this stage of the method, the existing variants for each sub-process of the main process (or its corresponding sub-process at one level higher in the process hierarchy) is compared for the purpose of determining how similar or different they are from each other.
6. Construct the variation map – from the previous steps, the business drivers are present in the business process, the existing variants and their degree of similarity or difference are known. In this step, this information is used to determine if the variants of each sub-process should be modeled together or separately.
7. Assess Data Object and Resource Drivers – this optional step is only applicable if the process models are annotated with either data artifacts or resources or both. In case of process model discovery, this step would only apply if the scope of process modeling includes these elements. In this step, the data object and resource induced variability is examined. For each case where there are strong data object and/or re-

source induced variability, decisions on modeling the variants together or separately are taken.

The steps are performed concretely with business stakeholders who have in-depth knowledge and understanding of the business process or parts of it. The roles that possess this knowledge may vary from one organization to another. In some organizations it is the business analyst, in others it might be subject matter experts or process owners. The actual list of participants (and roles) is naturally determined in discussion with those who have request the work.

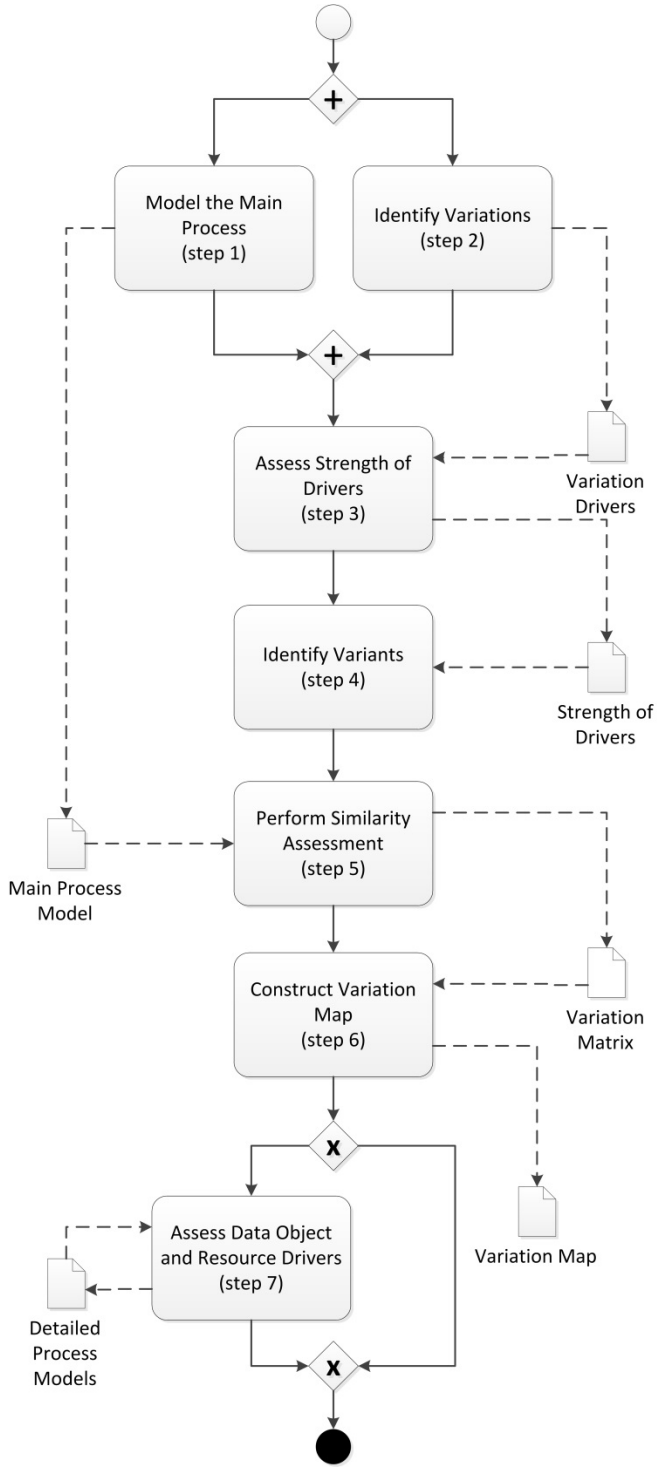


Figure 13: Steps of the Decomposition-driven method

5.3 Model the main process (step 1)

In this step, the main process is modeled. This is done together with the domain experts and other relevant business stakeholders. The aim is both to scope the business process in question and to identify the major (high level) milestones in the business process. The level of abstraction of the main process should be at such level as to depict the major 5±2 steps (sub-processes) of the business process. One possible method for modeling the main process is introduced in [167] by Sharp and McDermott. In this method, the start and end events are first identified and then milestones are discovered. Alternative methods are, among others, introduced by Dumas *et al* [54], Harmon [77], and Rummler and Brache [155]. Although these methods vary slightly in how the main process is elicited, they all provide the concepts necessary for modeling the main process and their further decomposition. As have been shown previously, different decomposition heuristics will affect the process hierarchy but not, in a significant way, their understandability.

One suggested way is applying the "breakpoint" decomposition heuristics is by following the steps described below;

1. Identify what will initiate the main process, i.e. when does the main process start (for instance, when a customer order is received)
2. Identify what will conclude the end process, i.e. when does the main process end (for instance, when the customer has paid the invoice)
3. Determine what major steps are needed to go from the start of the main process to its conclusion. A main process usually have 5±2 major steps (sub-processes) [167].
4. Organize the sub-processes in the order they are executed from the start of the main process until the end.

Note that the first and second step does not have to be conducted in this order, it is fully possible to do step 2 first and proceed with step 1. This should be decided depending on context. For instance, if the domain experts are familiar with process modeling or have already a main process modeled, then it is better to start with step 1. However, if the domain experts are unfamiliar with process modeling, it is better to start with step 2 so they get introduced to the concepts of processes and value creation.

In Fig. 14, the main process of a governmental agency managing applications for constructions is shown. It starts with a plan being received and then it is registered, prepared, examined and finally approved. As such, the main process represents the major milestones of the business process. The main process is needed for consolidating process models and discovery of process models.

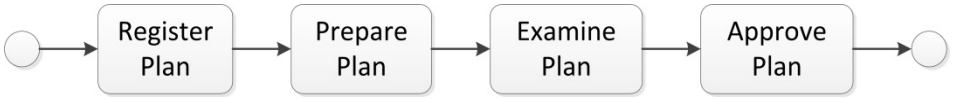


Figure 14: Example of a Main Process

5.4 Identify variation (business) drivers (step 2)

5.4.1 Eliciting Business Drivers

The second step is to elicit the variants (induced by business drivers) of the business process and classify them. For the elicitation of variation drivers, the adapted framework of Rummler and Brache (presented in the previous chapter of this thesis) is used (cf. Fig. 15). As previously clarified, this framework depicts an organization as a system operating within a larger system. As such, factors from the larger system affect the organization and therefore can cause variability in the business processes of an organization.

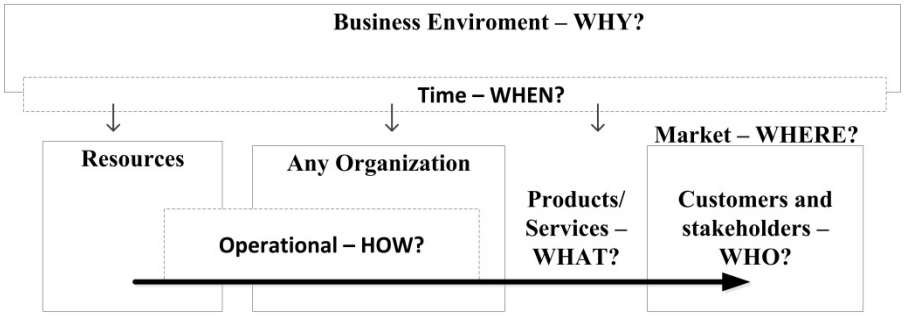


Figure 15: Framework for Variation Drivers

As the first step, the stakeholders of the organization are asking two rounds of questions in relation to the adapted framework depicted above. In the first round, questions are asked about the existence of drivers in each of the categories of the framework. The main questions for each of the w-questions are as follows;

1. What products/services does the main process produce?
2. Who are the customers/consumes (segments) of these products?
3. Where are these products distributed?
4. How are these products/services produced?
5. Are there any external circumstances that require a different process for meeting the specific needs and if so, when (for more clear examples, see previous chapter under time variations)?

These questions serve as a starting point but naturally, there will be follow up questions for the purpose of elaborating and/or adapted to the specific context.

In the second round, each of these categories of drivers are further clarified and refined (such as how many sub-segments of customers are served in this process). For instance, in the example shown in Fig. 17, the first round of questions identified the existence of product drivers (new construction request or change to an existing construction). In the second round, the discussions clarified that new construction plans could be for either office or residential buildings. Concretely, this is achieved by means of a workshop or interview with business stakeholders. The output of this step is a list of all possible variation drivers for the business process and therefore, implicit branching points.

5.4.2 Identifying and Classifying Viable Variants

As previously described (in the previous chapter), an explicit or implicit branching point is either defined as a decision or as a variation point. If it is a variation point, the outgoing branches are seen as viable variants. As such, in order to identify viable variants, the variation points need to be identified first.

The viable variant elicitation (as depicted in Fig. 16) begins with identifying all branching points of a given process model. In case of process model consolidation, both explicit and implicit branching points are investigated. In case of process model discovery where there are no process models, all branching points will be implicit. From the previous step, eliciting business drivers, the implicit branching points are identified (both for process model consolidation and discovery).

Once a branching point has been identified, the outgoing alternatives are examined to assess if they lead to different but similar outcomes, that is, classification of the branching point as either decision or variation point.

A viable variant is defined as "the outgoing path from a design-time variation point, when the different outgoing paths have similar inputs and/or lead to similar outputs as perceived by a domain expert". In determining of a branching point is a variation point or a decision point, the following questions might be of assistance;

- Does the variants have identical or similar starting events?
- Does the variants produce identical or similar outcomes?
- Is it a design-time variation, i.e.
    - Has the branching point been defined at design time and/or
    - Is the execution path of the process determined at the start of the process (as opposed to, is the execution path of the process determined just before the branching point)?
- Are the variants closely related to each other, i.e.
    - If the two (or more) variants were to be represented at one level higher as a sub-process in the process model hierarchy, would it be reasonable to include these variants in one sub-process?

- Does these variants belong the same business driver category (product, customer, market, operational or time)?

If the answers to the questions above are "yes", then it is defined as a viable variant.

Once a variation point has been identified, its variation options are identified, allowing identification of its variation driver. Continuing the analysis, identification of which W-question corresponds best to the variation driver is done and they are classified accordingly. The task beginning from classify branching points to classify variation driver, are repeated for each branching point in the collection of process models. It should be noted that in some cases, certain variants might be known before the analysis start, and in other cases the variants are discovered during the variation elicitation analysis. It might even be a combination of these two, that is, some variants are known at the start and some are discovered during the analysis.

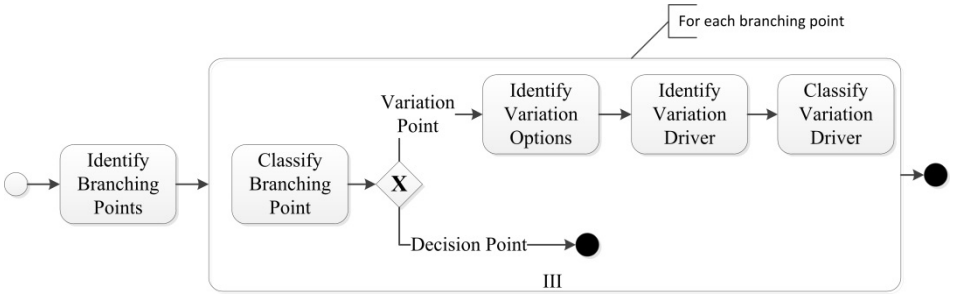


Figure 16: Driver elicitation method process

When analyzing a collection of process models, different analysts might choose to focus on different aspects or levels of granularity and thus recognize different variants in the process (due to for instance the purpose of the analysis work). This method does not provide the analyst with a prescriptive definition of what constitutes a process variant but rather guidelines to assist in the elicitation of viable variants. It will be the choice of the analyst to determine what constitutes a viable process variant. For example, the analyzer will choose whether to treat the processes for handling personal, vehicle and property claims as three different variants or a single process.

5.5 Assess the relative strength of variation drivers (step 3)

Having identified the viable variants and their business drivers in the business process under examination, a rating of importance (hereby called *strength*) is

assigned to each driver. The strength of a variation driver reflects the level of investments needed to merge or standardize the process variants induced by the driver, as well as the level of management such decisions would be made. Importantly, the aim in this step is to rate the business importance of each variation driver, regardless of how much the variants differ from one another.

Variants induced by a "very strong" driver are fundamental to the business. For instance, if a company provides a service in two different markets, each with different regulations, the market driver is considered as "very strong". It would be very difficult (if even possible) to make changes in the variants across markets (such as standardizing the two variants). Similarly, a product driver would be rated as "very strong" if a decision to substantially change the way one of the products is delivered would be seen as a change in the business model and would require a decision at the highest level of management. In other words, a very strong driver is such that its induced variants must be managed separately.

On the other hand, variants induced by a "strong" driver can in principle be merged or standardized to the point they can be managed together. However, their merging or standardization requires significant investments and decisions from mid-to-upper management layers. For example, consider a company that has two different IT systems to support the same service due to historical or organizational reasons. A decision to merge or replace these two IT systems would require significant investment but would not affect the business model. Changes in variants induced by "strong" drivers are confined to individual business units and require decisions from the management of the business unit in question.

Variants generated by drivers rated as "somewhat strong" are considered to differ only at the level of minor details from a business perspective. In other words, whether these variants are managed together or separately is irrelevant from an upper management perspective. Change decisions on variants induced by a "somewhat strong" driver are taken at a low or mid-management tier.

Finally, a driver is rated as "not strong" if it is irrelevant to the business whether the variants are merged or kept separate, or in the latter case whether they are managed together or separately. For example, consider a company that provides the same service to two or more customer segments where differences in customer segments do not play a significant role in the way a service is sold and delivered. In this setting, the driver "customer segments" can be rated as "not strong".

The rating of drivers can be achieved via a workshop or interviews with domain experts, based on the questions outlined in Table 15. For a given driver, the first question to which the answer is positive determines driver's strength rating. If the answer to every question is negative, the variation driver is rated "not strong".

Only the strongest driver is taken as primary. If multiple drivers have equal strength, the one with fewer sub-categories is ranked higher. For instance, if an insurance company considers its products (individual travel insurance and busi-

ness travel insurance) to be of equal strength as its customer drivers (northern, eastern, southern and western market segment), the product driver is ranked higher. The product driver has only two sub-categories whereas the customer driver has 4 sub-categories. In such cases, the driver with fewest sub-categories is ranked higher so as to reduce duplicity in the variation matrix.

In the running example, the primary variation driver was to be identified as the product driver with "new construction" and "change construction" plan. Having rated the relative strength of the variation drivers, this data is used to populate the first column of the variation matrix. The output of this step is a list of variation drivers for the process under examination together with their strength rating.

Table 15: Questions to help Determine the Strenght of a Driver

Rating	**Question**
Very Strong	Would a merger of the variants due to this particular variation driver be possible? Would a merger of these variants affect the business model or structure in such a fundamental way that it would require a decision from the highest level of management?
Strong	Would a merger of the variants (if desirable) require considerable investment, including noticeable re-organization, and require decision from high level of management?
Somewhat Strong	Would a merger of the variants (if desirable) require some investment, include some re-organization noticeable to the concerned business unit only, and require decision from mid-level management?
Not Strong	None of the above.

5.6 Identify the variants of each sub-process (step 4)

In the fourth step, existing variants for each sub-process (as identified in step 1) and for each variation driver are identified. This is concretely done by asking the business stakeholders for each sub-process, existing variants per business driver and adding them, one by one, to the variation matrix. The variants are therefore captured in a textual way by their name. The output is a variation matrix (cf. Fig. 17) wherein the rows correspond to business drivers (qualified by their relative strength) and the columns correspond to the sub-processes identified in step 1. A cell in this matrix lists the variants of a given sub-process (if any) induced by a given driver. For convenience the drivers are listed in descending order of strength. For instance, in the running example, there are three

different variants for examining a plan (examine a new construction plan for office building, residential building and change to existing construction).

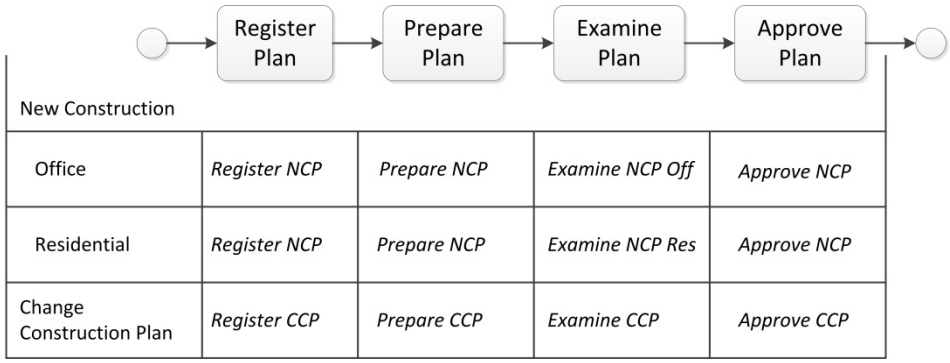


Figure 17: Variation matrix

5.7 Perform similarity assessment of variants of each sub-process (step 5)

In this step, a similarity assessment for each subset of variants of each sub-process identified before is performed. As discussed previously, for this similarity assessment it is not assumed that detailed models of each sub-process are available for comparison. Naturally, if such models exist, they can be used. Accordingly, a 4-point scale for similarity judgments extensively used in the field of similarity assessment [196], is employed: (1) very similar, (2) similar, (3) somewhat similar, and (4) not similar. It should be noted that identical variants are marked as identical and not subjected to similarity assessments.

This step can be implemented by interviewing the domain experts, asking them – given the identified variants of each sub-process – if the variants of the sub-process are likely to lead to models that are identical, very similar, similar, somewhat similar or not similar (cf. Table 16). The output of this step is an annotated variation matrix, where is set of variants of a sub-process is annotated with the result of their similarity assessment.

For instance, in Fig. 17 there are two variants of register plan (register NCP and register CCP). The business stakeholders are asked about the similarity of these two variants. Color codes or any other method of choice can be used to distinguish the similarity of the sub-process variants. If more variants are available, such as in the case of "examine plan" in Fig. 17, the same procedure is repeated but beginning with variants within one variation driver first. For instance, the similarity of "examine NCP Off" is compared with "examine NCP Res". Then, they are assessed as compared to "examine CCP".

Table 16: Guidelines for Subjective Assessment of Similarity

Similarity Assessment	Similarity of two variants
Identical	There is no perceivable difference.
Very Similar	Differences can be perceived but they are not significant.
Similar	There are clear similarities throughout the process.
Somewhat Similar	There are some isolated parts of the process that are perceivably similar.
Not Similar	There are in essence no perceivable similarities

5.8 Construct the variation map (step 6)

From the previous steps, the strength of the business drivers and the degree of similarity between the variants of each sub-process induced by a driver are known. This information is used to assess the trade-off of modeling the variants in a consolidated versus fragmented manner. In making a decision to consolidate or fragment, the analyst will use the decision matrix depicted in Fig. 18.

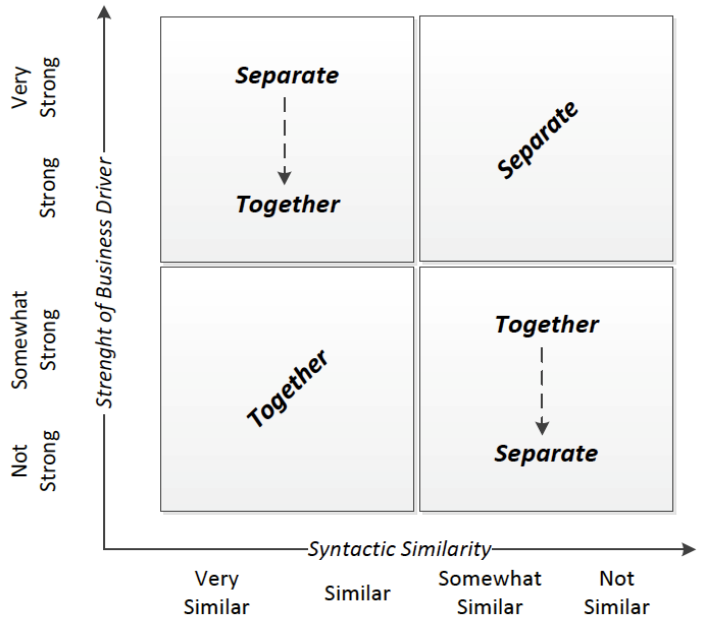


Figure 18: Decision matrix for modeling variants separately or together

If the variants are very similar and there are no strong business drivers for variation (not strong or somewhat strong i.e. no significant business impact), then naturally the variants are modeled together. Conversely, if there are strong business drivers (strong or very strong i.e. they have business impact) and the variants are syntactically different (somewhat similar or not similar) they are modeled separately. If variants are similar and have strong business drivers, they are modeled together or separately depending on the current level in the process decomposition. By high level of decomposition, level 3 (level 1 and 2 refer to Business Model and the main process) of the value creation system hierarchy introduced by Rummler and Brache [155] is referred. Using the same process architecture, low levels of decomposition refer to levels 4 and 5 (lowest levels of decomposition).

At levels close to the main process, sub-process variants falling in this quadrant are modeled separately because the business driver for separating the variants prevails. If the business driver is strong, it pre-supposes that the variants have different process owners and stakeholders and therefore the modeling effort has to be done separately for each variant. At lower levels of process decomposition, the business driver for modeling two variants separately weakens down and the incentive for sharing the modeling effort for variants increases. Therefore for sub-processes at lower levels of decomposition, the syntactic driver prevails, i.e. if these processes are similar, they are modeled together as a consolidated sub-process. Conversely, in the lower right quadrant, variants of sub-processes at a high level of decomposition are modeled together, since these variants fall under the same ownership area and thus it makes sense to conduct a joint modeling effort for them. However, at the lower levels of decomposition, if two sub-process variants are not similar, the analysts can choose to model them separately.

The output of this step is a variation map (cf. Fig. 19). A variation map is a process model where there are only tasks and XOR splits, representing the points where multiple variants will be separated. In constructing the variation map, only the allowed combinations are modeled using gateways. As such, the variation map shows the variants of each sub-process that ought to be modeled separately. The variation map contains one decision gateway per subset of variants of a sub-process that need to be modeled separately. If a sub-process does not have variants, it is not preceded by a gateway.

For instance, having performed the similarity assessment based on the variation matrix in Fig. 17, it has become known that variants “Examine NCP Off” and “examine NPC Res” are very similar to each other but different from “Examine CCP”. As such, as it concerns a high level of decomposition, the variants for NPC are modeled together and CCP is modeled separately from NCP as according to the decision matrix in Fig. 18. Furthermore, when constructing the variation map, constraints between variants of pre- and succeeding sub-processes are considered. For instance, as depicted in Fig. 19, only “examine NCP” can follow “prepare NCP”, i.e. it is not possible to execute “examine CCP” after “prepare NCP”.

Having constructed the variation map for the first level of process decomposition, each of the sub-process variants in the variation map, are considered in turn. Each sub-process variant is then decomposed into a lower-level process model and steps 2–4 are repeated at this lower level.

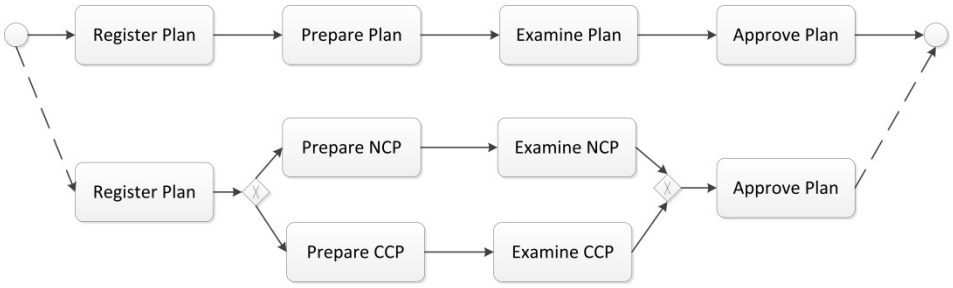


Figure 19: Variation Map

5.9 Modeling the Business Processes

At this stage of the implementation of the method, the processes are to be either re-modeled (process model consolidation) or modeled from scratch (process model discovery). When process modeling is done for discovery purposes, there are two main approaches one can choose from, 'bottom-up' or 'top-down'. The decomposition driven method presented here is compatible with both approaches but it is recommended to apply the bottom-up approach for process model consolidation and top-down approach for process model discovery.

In case of process model consolidation, the variation map has already outlined which processes are to be modeled together and which are to be modeled separately. In such case, it is recommended to adopt a bottom-up approach (beginning with the process models at the lowest level of hierarchy) and "dice" the process models in accordance with the variation map. As the fragmented models are available, it is feasible to take advantage of the work already done and re-arrange and re-model them in accordance with the outline given by the variation map.

In case of process model discovery, the main process, the main variants of each sub-process of the main process and their sub-variants are already elicited. As such, it is feasible to continue the modeling work along the same path to the lowest level of granularity by modeling the activities of the sub-processes.

The decomposition driven method does not depend on any particular requirements as to the modeling techniques used. Therefore, the modeling itself that is done together with the domain experts, can be done in accordance with the guidelines provided by for instance Dumas et al, [54], Harmon [77] or Sharp and McDermott [167].

5.10 Data Objects and Resource Driven Variation

A business process may also have variability induced by its data objects or resources. This is only visible if the process model has been modeled at the lowest level of decomposition. The lowest level of decomposition is that level at which input and output data objects are modeled, as well as the performers of the activities (for instance by using pools and lanes in BPMN notation). Previous steps of the method introduced, have concerned modeling processes together or separately, but at this step, activities within a process model in focus.

In other words, decisions on whether to model activities of a process model in a consolidated or fragmented manner, is determined for each task at the lowest level of decomposition. Such decisions are taken on the basis of the "strength" of the data objects and/or resources, and the degree of similarity in the underlying procedures of that activity. Previous steps have been concerned with the "major variations" of a family of process models as they have been on the level of process models (variants). In this step, the "minor variations" are addressed as the focus is on the activities of a process model.

This step can be applied on 'process model consolidation' and in 'process model discovery'. As before, a consolidated modeling approach for each activity or task should be the default option until it becomes clear that a fragmented approach is preferable from a data object and/or resource perspective. These ideas are embodied in the following steps.

5.10.1 Elicit Data Objects and Resources.

The first step is to elicit the data objects and resources performing the tasks for each activity of a process model. This is achieved by, for each process model, listing the activities it is composed of. If the process model includes a sub-process, then the same procedure is applied on the expanded sub-process. If a sub-process does not merit the effort of being decomposed, the same steps are applied as for activities. However, the difference is that the steps elaborated below, are applied on the sub-processes as opposed to on each activity of a sub-process. This information is then entered in a template as illustrated in Fig 20.

Process Model A	Input Data Objects	Output Data Objects	Resources
Sub-process 1			
Sub-process 2			
Sub-process 3			

Figure 20: Template for Data Object and Resources

Once the list is completed, by means of workshops or interviews with domain experts, the data objects and resources that cause variability are highlighted. Note that only those activities that have variability in data objects (either input or output or both) and/or resources are of interest. In order to know this, the data objects and resources of each activity are noted, but one can choose to enter data for only those activities that have variability.

A data object does not cause variability if it is always required or produced. However, if there is a data object used in a certain case and not in another, i.e. one data object exclude the use of another, there is a potential cause of variability. Similarly, if an activity is performed by only one resource, there is no variability. However, if two or more resources perform the activity depending if a specific condition is fulfilled, it can be a cause of variability. At this stage, the template will have highlighted all sources of potential variability caused by data objects and/or resources.

5.10.2 Design-Time and Run-Time Variations

Next, a second filtering takes place. During this second filtering, all variations occurring at run-time [135], are excluded. The template will only include build-time [135] (a.k.a design-time) variability caused by data objects and/or resources (as defined in previous chapter). At this stage, all the data objects (input and output) associated with each sub-process and tasks, those that cause variability and specifically, those that cause build-time variability are known. Next, as described below, the strength of the drivers for variability are assessed.

5.10.3 Assess Strength of Drivers for each Activity

As input from the first step, a list of all data objects and resources that cause design-time variability is available. The next step is to assess the "strength" of the data object and resource driver for variability. This is achieved by assessing the data object driver followed by the resource driver. For both data objects and resources, a two-point scale consisting of "strong" or "not strong" is used.

The importance or "strength" of a data object as driver for variability is dependent on if the variability is on the "object", "object type" or "attribute" level. Variability in "objects" and "object type" are stronger than "attribute". For example, if an activity uses different objects (purchase or customer order), the data object driver is considered as "strong". Similarly, if an activity has as input, different types of customer orders (such as corporate or private customer order), the variability driver of the data objects is considered to be "strong". However, if an activity uses the same type of customer order but the attributes (such as which fields of the order is required to be filled in) are different, the data object driver is considered to be "not strong".

The strength of resource driver for variability is dependent on if the variability occurs at the level of "business units", "sub-divisions" of a business unit or

"roles" of a sub-division (see previous chapter). The resource-induced variability is "strong" if an activity is performed by two different business units (if certain conditions hold) or two different sub-divisions (if certain conditions hold). Similarly, the resource driver is considered to be "not strong" if there is a variability in roles within a sub-division.

The output of this step is a documented assessment of the relative strengths of the data object and resource drivers for variability for each activity of a process.

5.10.4 Assess Similarity of Variants for each Activity

In this step, a similarity assessment is performed for each activity that has variability caused by either data objects or resources. As discussed previously, for this similarity assessment, it is not necessarily to have detailed models of each sub-process. The scale for similarity judgment is subjective and follow the same structure as mentioned above and extensively used in the field of similarity assessment [196]: (1) very similar, (2) similar, (3) somewhat similar, and (4) not similar. (Identical variants are marked as identical and not subjected to similarity assessments).

This step can be implemented by interviewing the domain experts, asking them – given the identified variants of each sub-process – if the variants of the activities (and in some cases, sub-processes) are likely to lead to models that are identical, very similar, similar, somewhat similar or not similar (cf. Table 16). It should be noted that the procedures of the compared activities are assessed for similarity. The output of this step is an annotated template (introduced in step 1), where each set of variants of a sub-process is annotated with their degrees of similarity. As before, color codes or any other method of choice can be used to distinguish the degree of similarity of the activity variants.

5.10.5 Determine how to Model each Activity (consolidated or fragmented).

From the previous steps, the strength of the data object and resource drivers and the degree of similarity between the variants of activities, are known. This information is used to assess the trade-off of modeling the activities in a consolidated versus fragmented manner. In making a decision to consolidate or fragment the variants of a particular activity, the decision matrix depicted in Fig. 21 is used.

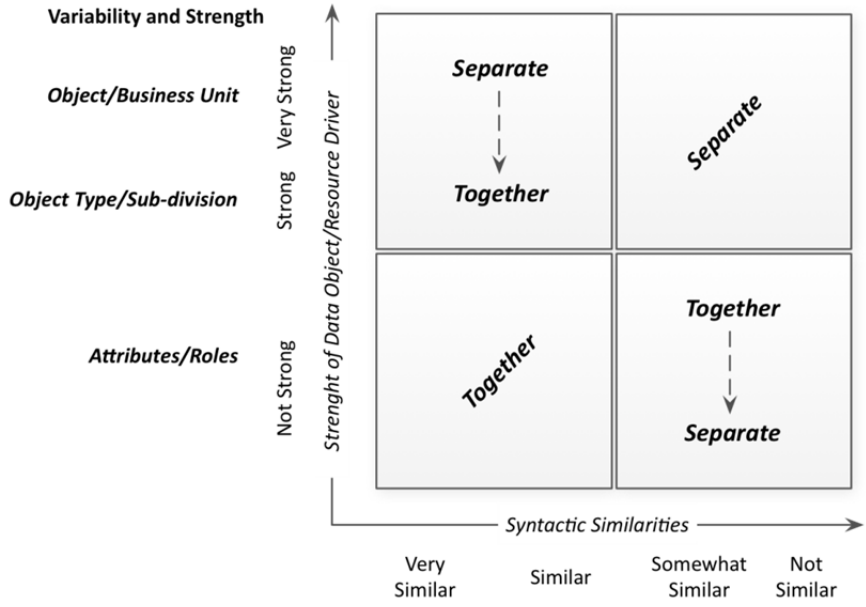


Figure 21: Decision matrix for modeling sub-processes and/or tasks separately or together

If the variants are very similar and there are no strong drivers for variation (not strong), then naturally the variants are modeled together. Conversely, if there are strong drivers (strong) and the variants are syntactically different (somewhat similar or not similar) they are modeled separately. If variants are similar and have strong drivers, they are modeled together or separately depending on the level of process decomposition. If it is a sub-process at a higher level of decomposition, it is modeled separately but if it is an atomic activity, i.e. no further decomposition is meaningful, it is modeled together. Sub-processes, as they are of higher order in the level of decomposition, are modeled separately as the driver for separating the variants prevails. If the data object and/or resource driver is strong, it pre-supposes a significant reason for variability, expressed for instance as using different objects, and therefore the modeling effort has to be done separately for each variant. At the lowest level of process decomposition, the drivers for modeling two variants separately weaken down and the incentive for sharing the modeling effort for variants increases. Therefore for activities at lower levels of decomposition, the syntactic driver prevails, i.e. if these processes are similar, they are modeled together as a consolidated sub-processes or activities. Conversely, in the lower right quadrant, variants of sub-processes are modeled together, since no significant business reason drive variability. As such, it makes sense to conduct a joint modeling effort for them. However, at the lower levels of decomposition, if two activities are not similar, the analysts can choose to model them separately.

If a sub-process or activity has data object and resource induced variability at the same level of strength (strong versus not strong), the similarity perspective will determine the modeling. Similarly, if they are in the same level of similarity (very similar and similar versus somewhat similar and not similar), the strongest driver determines the modeling choice. However, if a sub-process or an activity, for instance, has data object variability that is strong with variants that are very similar or similar, and a resource variability that is not strong with variants that are somewhat similar or not similar, the variants will be modeled separately. At higher levels of decompositions (sub-processes), determinant will be the driver perspective and at lower levels of decomposition (activities), the syntactic perspective takes precedence.

The variability induced by data objects can be captured even if sub-processes or activities are modeled together. One option is to capture the information in textual form separately from the process model, either at the attribute level (such as InputSets and OutSets of BPMN) [118] or in separate text documents. Such methods, however, do not allow visual expression of the data object variability. In addressing this issue, notation can be used to visualize the variability. For instance, La Rosa [149] propose notational extension of C-EPC to enable representing data objects involved in the performances of tasks (called C-iEPC). Although their contribution is used for configurable process models (C-EPC), it can be transposed to BPMN (only for data objects). Adopting the transposition of C-iEPC to BPMN to extend the use of control flow gateways to data objects, will allow for representing data object variability in process models. It would be sufficient to use exclusive (XOR), parallel (AND) and inclusive (OR) gateways to represent data object variability of sub-processes or tasks.

6 CASE STUDIES

Previous chapter presented a method for striking a balanced trade-off between modeling each process variant separately versus collectively. The novelty of the method is in its core idea, namely to incrementally construct a decomposition of the family of process variants into sub-processes, determining if each sub-process should be modeled in a consolidated manner (one sub-process model for all variants or for multiple variants) or in a fragmented manner (one sub-process model per variant). This decision, as described, is taken based on two parameters: (i) the business drivers for the existence of a variation in the business process; and (ii) the degree of difference in the way the variants produce their outcomes (syntactic drivers including data objects and resources).

The applicability of the method is tested on two separate cases that share the need of having their business process variations managed but differ in terms of, for instance business model, size, context and process maturity. For this purpose, the case study method is applied.

The rest of this chapter is structured as follows. The case study method is introduced in Section 6.1 and the rationale of the case study in Section 6.2. The research question of the study is presented and discussed in Section 6.3 followed by presentation of the design of the study in Section 6.4. The execution is presented in Section 6.5 followed by the finings in Section 6.6. Finally, this chapter ends with discussion on threats to validity in Section 6.7.

6.1 Method

There are various definitions on what a case study is. Despite the many definitions, most of them include the concepts of *examining*, *inquiring* or *investigating* a *contemporary phenomenon* in its *natural setting* or within its *real-life context* [22, 144, 195] in a systematic or structured manner. In short, a case study can be defined as an empirical method that serves the purpose of investigating a certain reality within its real-life context [156], particularly when the boundaries between what is studied and its context are not clear [195].

Case studies are often used for exploratory purposes, but they are also suitable for evaluation of a method within the software and systems engineering domain [89]. The case study method can also be used for testing a hypothesis in, for instance a confirmatory study [55, 60, 156]. For confirmatory purposes, a hypothesis derived from the research question is either confirmed or rejected. A confirmation shows empirical data that is analyzed and interpreted in order to accept or reject hypothesis [55]. These features make the case study method applicable as an instrument to validate the proposed method.

A case study should include one or several explicitly stated research questions that is defined at the beginning of the case study [124, 157]. Furthermore, it is necessary to know what data to gather and plan for data collection in a consistent manner [124]. Using the collected data, inferences are made from the

data in order to answer the research question (and also accept or reject the hypothesis defined). There is also a need to discuss the threats to validity and present measures taken to reduce these risks [124].

6.2 Rationale

6.2.1 Case 1: Mid-sized European Bank

A mid-sized European bank offers trading services in foreign exchange (FX) and money market (MM). Foreign exchange is trading in currencies such as euros against dollars. For instance, an US based multinational company might receive orders from Europe with payment in euros. As the company wants to have its funds in dollars, it will approach a bank to sell the euros they have received and buy dollars instead. As such, they trade euros in favor of dollars and have therefore performed a foreign exchange transaction. Money market is when banks or larger corporations trade in loans and deposits that range from one day to considerably longer periods. Consider the same US based company. They might have received a large payment in dollars, which they will need in the following week. Instead of keeping the funds in their own accounts for a marginal interest rate, they can deposit the sum (lend it out to someone who needs short term funds) for a higher interest rate. As such, they have made a money market transaction. Conversely, if they would need funds for a short term, such as two weeks, they can take a loan from someone who has excess funds during this time period.

The bank in this case study (henceforth refereed to as the bank), offers these services of trading FX and MM transactions. For this purpose, the bank has a legacy system for its back office processing of FX and MM trades (families of process variants). The bank seeks to replace this system with an off-the-shelf system. Although the FX and MM market is one of the most mature financial markets, there are many developments made due to new regulatory requirements, customer demands, and adaptations to international standards. In order to continue providing FX and MM related services competitively, the bank needs to continuously develop enhancements in there IS system and structure. The legacy system of the bank has been caught in a “maintenance trap” (cost of necessary maintenance is large compared to development of new functionalities). Furthermore, as it is an old system (some 20 years), it is complicated to develop, it is difficult to find good developers who master the language in which it is written, and it takes a long time for a new resources to get enough aquatinted with the system before become productive developers. In addition, the bank is concerned about their competitiveness as the cost of maintenance is successively increasing.

Due to these reasons, the bank wished to evaluate alternatives in the form of standard off-the-shelf systems. However, during the past 20 years, the system has been tailored to their specific needs, as it has been tightly integrated with supporting systems for other products. Furthermore, the business units have set

up their organizational structure in such a way that several different units use the same system but in various ways and for different purposes. While the current processes have their advantages, the bank wants do avoid a "replication" of the current processes within the framework of a new system. In order to achieve this, the bank saw it necessary to elicit requirements for all variants, which primarily come from the business processes rather than from the current IS and organizational structure.

The business processes had, several years before this case study, been modeled as separate process models (flowcharts) by a team of consultants. The existing models were flat (no decomposition had been made) and "sliced" in accordance with the organizational structure at that time. Three of these models were for the variants of the business process related to trading FX and MM with interbank counterparts and one for non-interbank clients (those who do not have an account with the bank). The bank aims at consolidating these process models prior to requirements elicitation. By doing so, the requirements needed for evaluating standard off-the-shelf system will be based on requirements needed for managing the products rather than reflective of current IS structure or organizational units.

6.2.2 Case 2: DNA Core Facility Center

A national bio-bank (repository of biological samples such as blood sample) of a country has collected more than 50 000 samples from the population for genome research purposes. This Genome Center also has a "core facility" where they perform genotyping and sequencing of DNA samples. Genotyping is when the differences of the genetic make-up of an individual are compared with another individuals DNA or against a reference sequence (DNA). DNA sequencing, on the other hand, is the process by which the precise order of nucleotides within a DNA is determined and documented. This service is performed both for in-house purpose (own research) and as a service sold to other institutions and companies.

This process is a heavily chemical one with many steps that require the full attention of the lab technicians performing the work. Furthermore, advanced machines are required for the sequencing. It is very important that they follow protocols when sequencing data. If protocols are neglected, it can pollute the data, which can be very costly.

Currently, they use manual tracking with excel sheets as support. Although it is crucial to follow protocols, they do not have their business processes represented in any other form than text documents. These documents function as instruction manuals but cannot offer any tracking support for their genotyping and sequencing projects.

The core facility division has initiated a collaboration with a software development company to build a Laboratory Information Management System (LIMS) to address these shortcomings. In addition, they wish to analyze their

genotyping and sequencing processes in order to find improvement opportunities, gain understanding of costs at various steps of the process, enable scaling of projects (managing larger number of projects, samples), improve their planning and reduce lead time, and incorporate barcode tracking devices to follow the progress of their projects. In order to incorporate these functionalities, they need to capture their desired business processes and provide the software developers with requirements for their new LIMS system.

As they currently do not have any business processes modeled, they cannot initiate this analysis and therefore, they cannot elicit the requirements for the new LIMS system. As such, they need to model their business processes. Their business processes have variations, i.e. the protocols are different depending on for example choice of method when sequencing the DNA. The way these business process variations are represented will affect how the analysis is made and have bearing on how the tracking solution will be. However, it is not clear how to horizontally and vertically arrange and organize the business processes. Furthermore, the requirements on their new LIMS system will be affected by how they choose to design their business processes.

6.2.3 Case Study Settings

The case studies differ in industry sector, transaction volumes, level of IS maturity, level of modeling experience of the domain experts, and the purpose and context (cf. Table 17).

With respect to industry sector, the banking industry is highly commercialized and involves large number of customers with high degree of regulations to comply with. In the core facility case, it is predominantly a research institution and is dependent to large extent on funding.

As to transaction volumes, the first case study involves large number of transactions. The foreign exchange market is the most liquid of the financial markets, and it is not uncommon to see transactions in the order of tens of thousands per day. These transactions need to be entered, processed and sent off within minutes. Delays will result in interest rate compensations, which is costly. On the other hand, the core facility manages few projects at a time, taking anywhere form a couple of days to couple of weeks to initiate, process and deliver results.

Table 17: Differences of the Case Studies

Attributes	**Bank**	**Core Facility**
Industry	Commercial	Research
Volume	High	Low
IS maturity	High	Low
Process Modeling Experience	High	Low
Purpose	Process Model Consolidation for Evaluation of a System	Process Model Discovery for Building a new System

As to the level of IS maturity, the large number of transactions processed by the bank, required several highly integrated and automated IS. In the second case study, volumes are low and projects involve a high level of manual processing with almost no IS support.

Regarding modeling experience, the experts of the banking case study had at least 5 years of experience with process models, a set of process models to begin with and a dedicated department for business development with dedicated business analysts. This is contrasted by the domain experts of the DNA case that had not worked with process models at all and had no process models prior to the implementation of the case study.

Finally, as to purpose and context, the two case studies are very distinct. In the banking case, there was a set of process models that needed to be consolidated (bottom-up approach), whereas in the DNA case, the process models were discovered (top-down approach). In both cases, the resulting process models were to be used as input for requirement elicitation, but in the banking case, for evaluation of standard packages and in the core facility case, for development of a new LIMS system.

6.3 Research Question

In both case studies presented above, there is a need of managing the variations of the business processes. It is for this very purpose, managing variations of business processes that the method described in this thesis has been developed for. The objective with these case studies is the application of the method described for confirmatory [55, 62] purposes.

The overarching research question and its two sub-questions of the case studies are:

RQ: How can a family of process variants be modeled?

RQ 1: How can a family of process variants be consolidated in a manner that results in the usage of fewer activities and sub-process models?

RQ 2: How can a family of process variants be discovered in a manner that results in the usage of fewer activities and sub-process models?

In the context of the first case study, it matches RQ1 (consolidation of existing models of process variants) while the second case study matches RQ2 (from-scratch modeling of a family of process variants).

Yin [195] states that there is a need for developing a hypothesis. The purpose of this method is to manage variants process models that have less redundancy than a collection of fragmented models. Thus, the hypothesis is that:

Hypothesis: "when this method is applied on a family of process variants, then the same set of business processes can be represented using fewer activities and sub-process models than if the same was done using a fragmented approach."

Null Hypothesis: "when applying this method, the size of the family of process variants is the same or larger in terms of total number of activities and sub-process models compared to a fragmented approach."

These research questions are relevant given that reducing the number of activities, in particular duplicates, and sub-processes, lead to better comprehensibility of the process models [194], less duplicity and stronger linking of related sub-processes. This in turn, will reduce maintenance efforts and will also facilitate the analysis, comparison and implementation of process variants in a common IT system[72].

6.4 Design

6.4.1 The Cases and Units

In the case of FX and MM operations, at the start of the case study (the context), the business processes had been modeled in a fragmented way, meaning that most variation had been modeled separately. From a quick review of the process models, it was apparent that they contained duplicates and could perhaps be consolidated. This case (unit of the case study), therefore, is confined for consolidating the operational business process models for FX and MM trades.

The context of the core facility division of the Genome Centre, the sequencing division had no business process models (in diagram form). The only available information and knowledge about the business processes are in the minds of the staff and to some extent captured as text in protocols. From a quick review and conversations with the head of the sequencing division, it was clear that these processes have variations. This case unit is confined to the discovery of the business processes of sequencing.

Given the contexts and units described above, a holistic design is chosen. A holistic case study design is appropriate when (i) there are no logical subunits of a case and therefore, no obvious additional unites of analysis and (ii) the theoretical framework supporting the study is itself holistic in nature [157]. In this

case, there are two case studies with one unit in each, and therefore a multi-case study is conducted.

6.4.2 Case Selection Strategy

The following criteria were elicited for case study selection.

1. **Variation of Cases**: As it is a confirmatory case study covering consolidation and discovery of business processes, variability of cases is necessary. Variability increases the validity of results as the method is applied on different settings and therefore, the assessment of its applicability is stronger.
2. **Variability in Business Processes**: As the method manages family of process variants, the cases selected must have variability in its business processes. If lacking variability, the method cannot be applied.
3. **Variation in Industry**: It is important for the cases to be from different industries – the method aim at being applicable in all business processes that have variations, regardless of the industry. In order to validate its applicability across industries, it is better to apply it to cases from different industries.
4. **Variation in Context**: It is important that the contexts are different – the method is generic for managing variations in business processes regardless of context and as such, the context of the application should be different.
5. **Access to Domain Experts**: It is important to have access to the domain experts as the method aim at discover or consolidate process models that are meaningful for the domain experts. Furthermore, the method relies on the input of the domain experts and easy access to them is therefore necessary.

Given the above research question, case studies where families of process variants needed to be managed collectively were sought. Naturally, case studies where domain experts could be engaged were required, as this method heavily relies on their input. Finally, the case studies needed to allow for addressing both research sub-questions and that were representative of different modeling purposes, industry sectors, level of IS maturity and transaction volumes. Below the case study design is presented.

6.4.3 Data Selection

In the bank case, a list of 'knowledge areas' the resources needed to represent was prepared. In addition, the bank was asked to suggest additional persons they felt could contribute with relevant information. The business processes of the FX and MM case cover two organizational units. Two resources were secured from one of the units (one being the production manager), one person from the other unit, a business analyst who works with development for these

two units, and finally, one of the managers. In addition, access to support staff from IT support and development was granted. Furthermore, access to resources from other units was granted. They were not required to be dedicated to the project but were available for questions and clarifications. These people combined, covered all aspects of the FX and MM business processes.

In the case of the core facility, there is only one unit working with these processes. Three persons, all of whom were working with sequencing and genotyping, were secured for the case study. These resources covered all aspects of the business processes under consideration.

6.4.4 Data Collection

Data was collected through direct and independent methods. Direct methods, when the researcher is in direct contact with the interviewees and data is collected in real time [157], were employed by using interviews. Interviews were conducted, both in workshops with several participants (focus groups) and individually with key persons in both case studies.

All the interviews were conducted in a semi-structured manner. In accordance with semi-structured interviews, a set of prepared questions was used as basis for the interviews. The semi-structured interviews allowed further exploration of additional issues that arose during the conversations. The objectives of the interviews were descriptive (of the business processes) and explanatory (why the business processes are as they are).

Independent methods of data collection (independent analysis of available work artifacts) were also employed. In the FX and MM case study, the main artifacts were the available process models that were studies, analyzed and consolidated. In addition to these process models, work instructions, project documents and other documents available were included in the data collection. In the case of the core facility, independent data was collected from protocols (work descriptions).

These results were studied in detail and cultivated. Some of the parts were deleted, others added, some re-formulated, re-arranged and re-categorized. This process was conducted iteratively with analysis of the data artifacts available. The results of these analyses were a set of consolidated process models (FX and MM study) and a set of process models (for the sequencing study).

6.4.5 Data Analysis

The data (collections of process models for FX & MM, DNA sequencing and genotyping respectively), was analyzed by using the following measures:

- Number of process elements
- Number of sub-processes
- Duplication Rate
- Complexity of process models

6.4.5.1 Number of elements

Number of elements as defined in table 18 (also briefly discussed in Section 3.2.3), is a common way of measuring the size of the process models. By elements are meant events, gateways and activities in a process model. The larger number of elements, the larger is the process model, i.e. the higher number of events, gateways and activities. The input process models will be compared with the consolidated/discovered process models where variants have been managed according to the method described in this thesis.

Table 18: Number of Elements

Name	**Number of Elements**
Description	Total number of events, gateways and activities in a process model.
Entity	Process Models (Family of Process Variants)
Attribute	Size
Unit	Number (of elements)
Range	[0,1,2…n]

6.4.5.2 Number of sub-processes

Number of sub-processes also measures the size of the process models representing the family of process variants. It will be a simple comparison of the total number of sub-processes of the input models (in the case of FX and MM) compared to the consolidated process models. The core facility case study does not have a set of input process models to compare with, as it is a process discovery. Therefore, these business processes will be modeled by following the method described in this thesis and then modeled again in accordance with a reference method (the method proposed by Sharpe & McDermott which is described in more detail below). Given these two sets of process models, a comparison of the size of the process models can be made based on the number of sub-processes.

As Table 19 describe, the entity is the process models representing the family of process variants and the attribute being measures is its size.

Table 19: Number of Sub-processes

Name	**Number of Sub-processes**
Description	Number of sub-process occurrences in the process models
Entity	Process Models (Family of Process Variants)
Attribute	Size
Unit	Number of sub-processes
Range	[0,1,2…n]

6.4.5.3 Duplication rate

The measures defined so far, measure the size of the process models. However, duplication rate measures the rate of which one activity occurs in several process models. If one activity occurs in two process models, it is counted as 2. If it occurs in three separate process models, it is counted as 3 and so on. The duplication rate is then the sum of duplicate activities divided by the total number of activities in all process models. The higher the duplication rate is, the more often the same activity occurs in the process models. This in turn means that the process models are larger than they need to be.

Table 20: Duplication Rate

Name	**Duplication Rate**
Description	Duplication of the same activities in process models.
Entity	Process Models (Family of Process Variants)
Attribute	Duplication
Unit	Ratio (duplicates per total number of activities)
Range	[0, ∞]

6.4.5.4 Complexity measure

Duplication rate measures duplicity but not the complexity of process models. For this purpose, Coefficient of Network Complexity (CNC) metric can be used (see Table 21). It measures the complexity of the process models. CNC is the ratio between the number of arcs and the number of nodes (elements). This simple metric has been put forward to be suitable for assessing the complexity of process models [2]. This CNC measure will be used both individually and collectively for the process models. The higher complexity a process model has,

the higher number of arcs (connectors) will there be to connect the elements. It is possible to create flat and thus very simple (non complex) process models but that would result in high duplicity of activities (as activities are duplicated and used in several process models). However, when consolidating or discovering process models, duplicity of activities is reduced at the expense of complexity. In order to represent the same process models, more arcs (connectors) are needed and as such, the ratio of arcs to elements increases.

Table 21: Complexity Metric

Name	**Complexity**
Description	CNC measure i.e. ratio of number of arcs and nodes (events, gateways and activities)
Entity	Process Models (Family of Process Variants)
Attribute	Complexity
Unit	Ratio (arcs to nodes)
Range	$[0, \infty]$

6.5 Execution

The design of both case studies (cf. Fig. 22) consists of nine steps, out of which the first eight steps correspond to the steps in the method introduced in the previous chapter. The final step, the ninth that does not have a corresponding step in the method as described previously, consists of verifying (with/by the domain experts) the process models that have been produced.

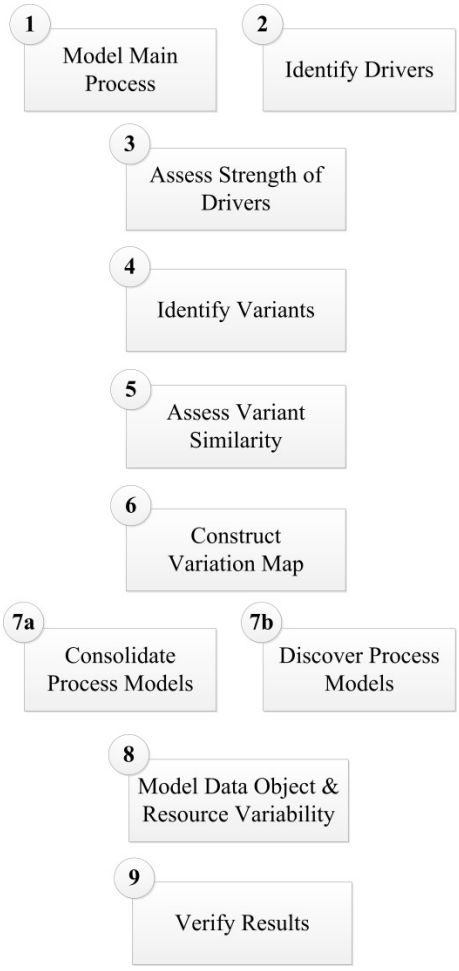


Figure 22: Case Study Design

There are two slight differences in the execution of the case studies. The cases, as previously explained, have differences that require slightly different implementations of the method. These are as follows.

(1) The business processes of the banking case study had already been modeled whereas in the core facility case study, had no process models to begin with. In the bank case, the main process was modeled again but verified and compared to the existing one. In the core facility case, the main process was modeled from scratch.

(2) The domain experts for the banking case study had at least 5 years of experience with process management and work with processes on a daily basis. However, the domain experts of the core facility case study did not have any

experience with process models. They had neither modeled nor worked with process models.

In the banking case (the first difference) the first step was to model the main process and then identification of the variation drivers. However, with the core facility case, the variation drivers were first identified and then the main process was modeled. This was due to the lack of experience of the domain experts in the core facility case. As a gradual introduction to the concept of main process, it was more meaningful to start with identifying the variation drivers and then model the main process.

Secondly, in order to examine the effectiveness of the method for the second case study (core facility), a baseline scenario or baseline process models (comparable to the input process models in the banking case study) was required for comparison. For this purpose, the business processes of the core facility were modeled according to the method presented in this thesis and according to guidelines presented by Sharp & McDermott [167]. This approach was chosen as baseline as it is widely used and recommended for practitioners [109].

6.5.1 Execution of FX & MM Case Study

The banking case study was conducted as described above. The method was applied in a four-hour workshop with five domain experts, led by the author of this thesis. In addition, two stakeholders from IT support were available for questions and clarifications. The workshop resulted in a variation map of the business processes. The first five steps were conducted in one workshop and in total took 4 hours. The first step (modeling the main process) took less than an hour and the elicitation and classification of drivers also took less than one hour. The similarity assessment, with the aid of the variation matrix, took around two hours. Afterwards, the variation map was modeled, which together with its verification, took three hours. The actual consolidation of process models took roughly 80 man-hours to complete. Afterwards, the process models were updated according to the framework for data object and resource variability. This effort took an estimated 16 man-hours to complete. Finally, the domain experts, in a series of 8 workshops, verified of the consolidated process models, each taking 1.5 hours on average to conduct.

6.5.1.1 Step 1 – Model the main process of FX&MM trades.

In the first step, the main process for managing FX&MM trades (cf. Fig. 23) was modeled. This was achieved by first asking what initiates the process and then, through a series of questions, modeling each step of the process until the end. The purpose of each sub-process was also clarified. In addition, the way and how each sub-process adds value to the process was summarized. This step resulted in a model of the main process for FX&MM products (Fig. 23).

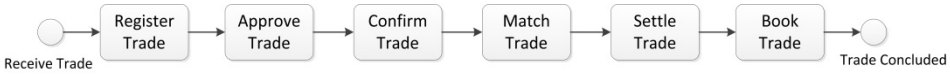


Figure 23: Main process for managing FX & MM trades

The main process is initiated once an order is received. The first task is to "*Register Trade*" meaning entering the trade in the IS. The next task is "*Approve Trade*". This is done automatically in most cases. Only the trades that fail an automated validation will require manual intervention and approval. Then, "*Confirm Trade*" takes place when the bank sends a confirmation of the trade details to the counterpart. Once the counterpart "*Match Trade*", i.e. agrees to the trade data, "*Settle Trade*" takes place (transfer of payment). The final task is "*Book Trade*" which is when the trade is booked in the accounting systems.

6.5.1.2 Step 2 – Identify the variation drivers

The second step was to identify variation drivers of the process. This was achieved by introducing the concept of variation drivers and the framework for classification. Following this, the introduction was made more concrete by showing some examples of variation drivers and asking the domain experts if their business processes have occurrences of such variation drivers.

Two variation drivers, product and customer driven variations, were observed to exist. The product driven variations (Fig. 24) were FX (foreign exchange), MM (money market) and NDF (non-deliverable forward i.e. trading in restricted currencies). The customer driven variations were identified as Bank (when the counterpart is another bank), Corporate (companies), Private (individuals) and Site (clients of various types that belong to one of the branches of the bank) clients. Furthermore, the corporate clients were of account (having an account agreement with the bank) or cash (do not have an account with the bank) client type.

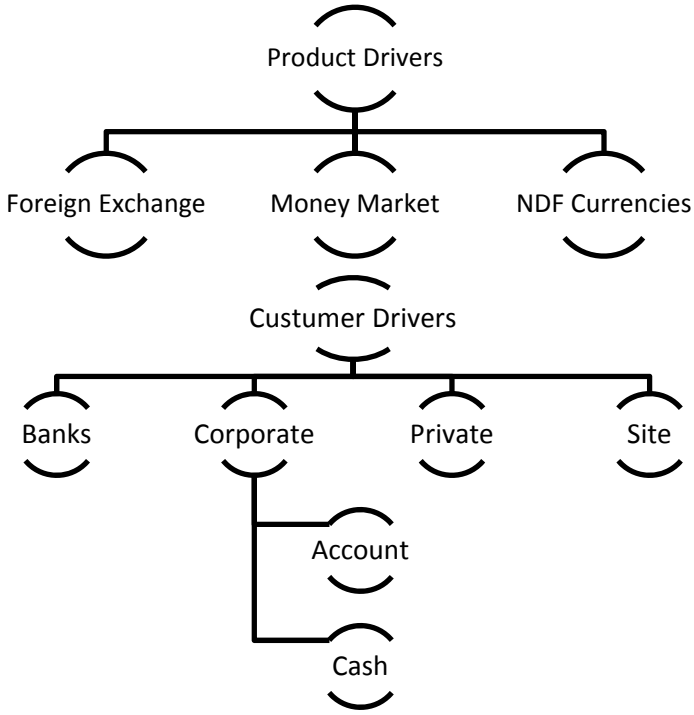


Figure 24: Variation Drivers for the Banking Case

6.5.1.3 Step 3 – Assess the relative strength of the variation drivers.

As input from the previous step, i.e. having the variation drivers identified, the next step is to determine their relative strength. Through discussions, it became clear that the product drivers were the strongest. It also became clear that FX & MM were similar enough to be treated as one. However, NDF is separate and on its own.

6.5.1.4 Step 4 – Identify the variants of each sub-process of the main process.

With the input from the previous steps, the variation matrix was populated. First, the variation drivers and their relative strength were used to populate the first column of the variation matrix. Then, for each sub-process of the main process, such as *"match trade"*, the domain experts were asked about how this process is performed? For instance, for an FX trade done with another bank, the ways to match are either Intellimatch (matching structure based on matching the data of confirmations sent and received by a IS system) or CLS (a centralized intra-bank platform created by the banks to serve as a third party in settling the FX and MM trades). As a result of the questions and answers, these two vari-

ants were entered in the matrix under sub-process *"match trade"* and for customer type *"bank"* (cf. Fig. 25). Note that in this case, the same solution (such as CLS) is used in "match trade" and "settle trade". As CLS is a centralized intra-bank platform, it has several functions and therefore used in two or more sub-processes. However, the use differs i.e. how CLS is used in "match trade" differs from its use in "settle trade". As such, in this case study, although the variants may bear the same name, they differ as they are situated under different sub-processes of the main process.

6.5.1.5 Step 5 – Perform similarity assessment of variants for each sub-process of the main process.

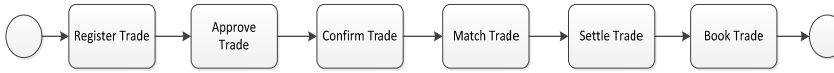
For the similarity assessment, each cell of the variation matrix was visited. In practical terms, this was achieved by asking the domain experts to

(1) Identify identical variants (if there were any) and then
(2) For non-identical variants, grade the level of similarity on a scale from 1 (very similar) to 4 (not similar).

For instance, the variation matrix shows that corporate and site clients have the same variants for matching a trade. By asking the questions stated above, the results showed that all SWIFT trades are very similar. The same applied to platform, online and paper. Furthermore, the domain experts assessed that swift, platform, online and paper are similar to each other as the process is basically the same but the tool used to match the trades differs depending on customer type. For instance, the processes for swift and paper are similar but differ only in what medium is used (swift or paper). It was also observed that matching in bulk (when several trades are matched at once) is very different compared to matching by SWIFT, platform, online and paper. As mentioned before, these variants are only compared to other variants under the "match trade" sub-process.

Having established the degree of similarity among the corporate, private and site clients, the next step was to ask about similarities between CLS and Intellimatch for when the counterpart is a bank. These differed significantly compared to how trades are matched for non-bank counterparts. This step resulted in identifying two main variants for matching when the counterpart is a bank (i.e. Intellimatch and CLS) and two main variants when trading with non-bank counterparts (i.e. when the matching is done in bulk versus single-trade match).

The same similarity assessment is performed for each cell of the variation matrix.



	Register Trade	*Approve Trade*	*Confirm Trade*	*Match Trade*	*Settle Trade*	*Book Trade*
FX & MM						
1. Bank	*Manual* *Automated*	*Manual* *Automated*	*Swift* *Online* *Paper*	*IntelliMatch* *CLS*	*CLS* *Gross*	*Gross* *Net*
2. Corporate						
Account	*Manual* *Automated*	*Manual*	*Swift* *Online* *Paper*	*Swift* *Platform* *Online* *Bulk* *Paper*	*Account*	*Gross*
Cash	*Manual* *Automated*	*Manual*	*Swift* *Paper* *CLS*	*Swift* *Platform* *Online* *Bulk* *Paper*	*Gross* *Net*	*Gross*
3. Private	*Manual*	*Automated*	*Paper*	*Paper*	*Account*	*Gross*
4. Site	*Manual* *Automated*	*Manual*	*Swift* *Online* *Paper*	*Swift* *Platform* *Online* *Bulk* *Paper*	*Gross* *Net*	*Gross*

Figure 25: Populated Variation Matrix for FX and MM

6.5.1.6 Step 6 – Construct the variation map.

By now, the variants for each sub-process of the main process are known and can be mapped into a variation map (cf. Fig. 26). For instance, there are two variants of *"Register Trade"* (manual and automated). These sub-processes did not have a strong business driver and were similar. Referring to the decision framework (Fig. 21), they are modeled together. Conversely, there are two variants of "*Settle Trade*" for bank clients in the variation matrix in Fig. 25 (CLS and gross). These were assessed to have a strong variation driver and also, to be 'not similar'. As such, in accordance with the decision framework (Fig. 21), they are modeled separately. After having continued in the same manner for all sub-processes, the variation map for each sub-process was constructed as depicted in Fig 26.

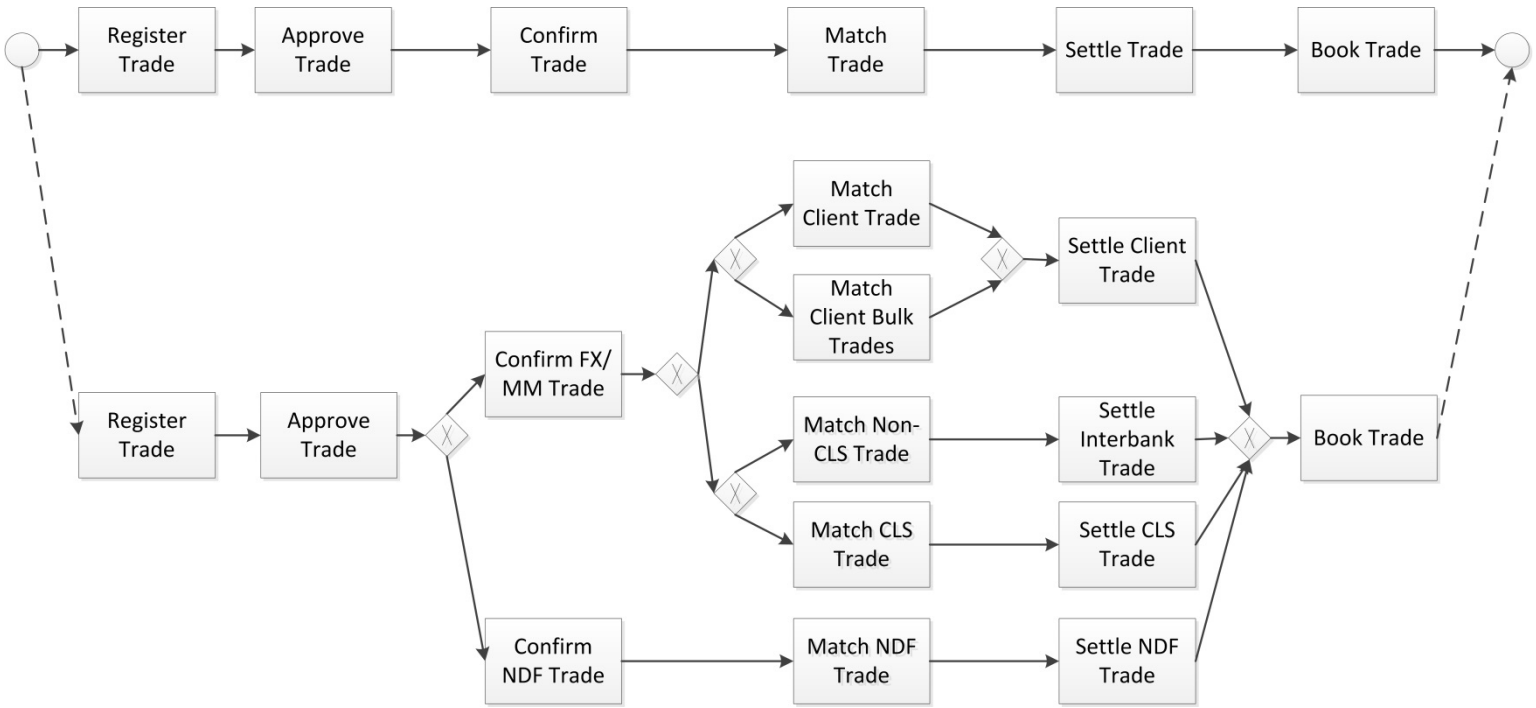


Figure 26: Variation map for FX&MM main process

6.5.1.7 Step 7a – Consolidation of Input Process Models.

The original process models had been modeled as flat end-to-end process models. As a first step, these models were diced or divided into sub-processes. This resulted in four process models:

(1) FX traded gross (traded and settled independently of other trades)
(2) FX traded via CLS (traded and settled using CLS)
(3) MM trades
(4) FX and MM for corporate clients

In addition to these four process models, there were two additional processes described as text, one for NDF and one for bulk matching, which was included in the models of this case study as part of the consolidation effort.

For each sub-process of the main process, the process models were consolidated in accordance with the variation map. For instance, the variation map states that there is to be one sub-process for "*register trade*". Therefore, the corresponding process models in the input process models were consolidated and merged into one model. The same procedure was repeated for the other sub-processes. Whenever clarification was needed, the domain experts and IT stakeholders were available for consultation.

It should be noted that the input process models had not been regularly updated with changes in the business processes during the past 3 years and therefore, minor discrepancies were observed. The consolidated process models were updated accordingly.

6.5.1.8 Step 8 – Model Data Object and Resource Variability

After the process models had been modeled, the models were revisited for elicitation of data object and resource variability. As previously elaborated, each activity with variability in resource or data object input or output, is examined to determine if that activity should be modeled as one or separated in two or more activities. This was achieved by first determining if an activity can be performed by only one or several actors. If only one actor performs it, then there is no resource-induced variability. However, if different actors performed an activity, depending on some criteria, the activity became a candidate for revision. The same logic was applied to data objects (both for input and output).

Following this, for each activity, depending on the degree of similarity of the detailed procedure of the activity and referring to the modeling framework introduced previously, each activity was either kept as it is or modeled as two or more activities. If the activity was kept as is, i.e. it was not modeled as two activities despite the presence of data object variability; the variability was captured through annotation. Activities with resource-induced variability were represented using pools and lanes.

6.5.1.9 Step 9 - Verify of Consolidated Process Models.

Once the process models had been consolidated, domain experts verified them in two rounds. In the first round, the lead domain expert (the one who coordinated all efforts) reviewed the process models and made adjustments. In the second round of validation, all the experts of their own particular area of the FX and MM processes verified the process models in a series of 8 workshops. The coordinating domain expert made adjustments to the consolidated models during these workshops.

After all workshops, the domain experts were asked about the usefulness of the models in terms of comprehensibility and if they would use the models for evaluating off-the-shelf systems. They stated that the consolidated models are easier to understand (compared to the input process models), and capturing more processes than the input process models. The consolidated process models were used as standard evaluation criteria for finding suitable replacement systems for supporting their FX/MM business processes.

6.5.2 Execution of DNA Case Study using the Decomposition Driven Method

For the core facility case study, three initial meetings were held with the head of the sequencing lab and two domain experts. During these meetings, information needed for constructing the variation map was gathered (steps 1-6). These steps took in total about 5 hours. Following this, the detailed discovery of the business processes was conducted (step 7b). Afterwards (step 8), the data object and resource variability was considered and process models were updated accordingly (requiring an estimated 40 man-hours to complete). Then, the domain experts verified the process models (step 9). The modeling effort itself (by the two researchers) amounted to circa 360 person-hours followed by 40 person-hours of verification by the domain experts.

6.5.2.1 Step 1 – Identify variation drivers.

For the core facility case study, no process models existed prior to the start of the study. Furthermore, the domain experts had no experience with process modeling. As discussed previously, the first step in this case study was to conduct the identification of variation drivers. This was achieved by introducing the concept of variation drivers, their classification and several illustrative examples for clarification purposes. Following this, the "W" questions, as discussed in the previous chapter on foundations of process variation, were asked in order to identify both stakeholders and uncover the variation drivers.

The analysis resulted in the elicitation of two variation drivers, product and operational drivers. The product driver is the co-existence of two distinct services: DNA sequencing (determining the order of nucleotides of sample con-

taining DNA) and array analysis (analyzing the genetic makeup of a DNA that determines specific traits). The operational driver relates to which machine is used for sequencing or analysis.

6.5.2.2 Step 2 – Model the Main Process of the DNA Sequencing Process.

The second step was to model the main process. By asking what triggers the process followed by which milestones the process goes through and what value each step produces, the main process was discovered. This led to the main process depicted in Fig. 27. The process is triggered when there is an agreement with a customer to sequence some samples. Then, the project data are registered followed by the samples being prepared. Once the samples are prepared, they are processed, meaning that they are sequenced using the sequencing or genotyped. In the final step, data is extracted, collected and delivered to the customer.

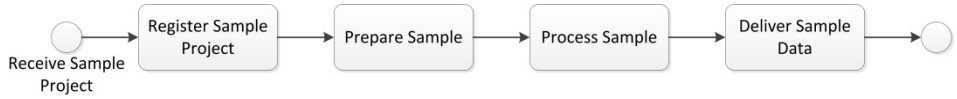


Figure 27: Main process for the core facility business.

6.5.2.3 Step 3 – Determine the relative strength of the variation drivers.

The questions in Table 15 were used to assess the strength of the variation drivers. This resulted in identifying the product driver (sequencing versus array analysis) as the strongest followed by the operational driver (HiSeq, MiSeq or Array Machine).

6.5.2.4 Step 4 – Identify variants of each sub-process.

Step 4 was executed in the same manner as the *banking* case study was done. This resulted in identifying two variants, namely "Prepare TrueSeq Sample" and "Prepare Nextera Sample", for sample preparation using HiSeq (cf. Fig. 28). One can also see from the variation matrix, that the same variants exist for preparing a sample when using MiSeq machine for sequencing. Similarly, it was noted that there are three different variants for Array analysis, one for processing DNA samples, one for RNA and one for Methylation.

	Register Sample Project	***Prepare Sample***	***Process Sample***	***Deliver Data***
Sequencing HiSeq	*Manual (Excel)*	*TruSeq* *Nextera*	*HiSeq Machine*	*FTP*
MiSeq	*Manual (Excel)*	*TrueSeq* *Nextera*	*MiSeq Machine*	*FTP*
Array Analysis	*Manual (Excel)*	*DNA* *RNA* *Methylation*	*Array Machine*	*FTP*

Figure 28: Populated Variation Matrix for DNA Sequencing

6.5.2.5 Step 5 – Perform similarity assessment of the variants for each sub-process.

The execution of step 5 also followed the same steps as with the banking case study. During the workshop, different colors of whiteboard pens were used to annotate the variants of each sub-process that were very similar, similar and so forth. This resulted in an annotated variation matrix.

6.5.2.6 Step 6 – Construct the variation map.

With the input from the previous step, the variation map could be constructed. It was constructed by deciding which variants should be modeled together and which should be modeled separately, resulting in the variation map depicted in Fig 29.

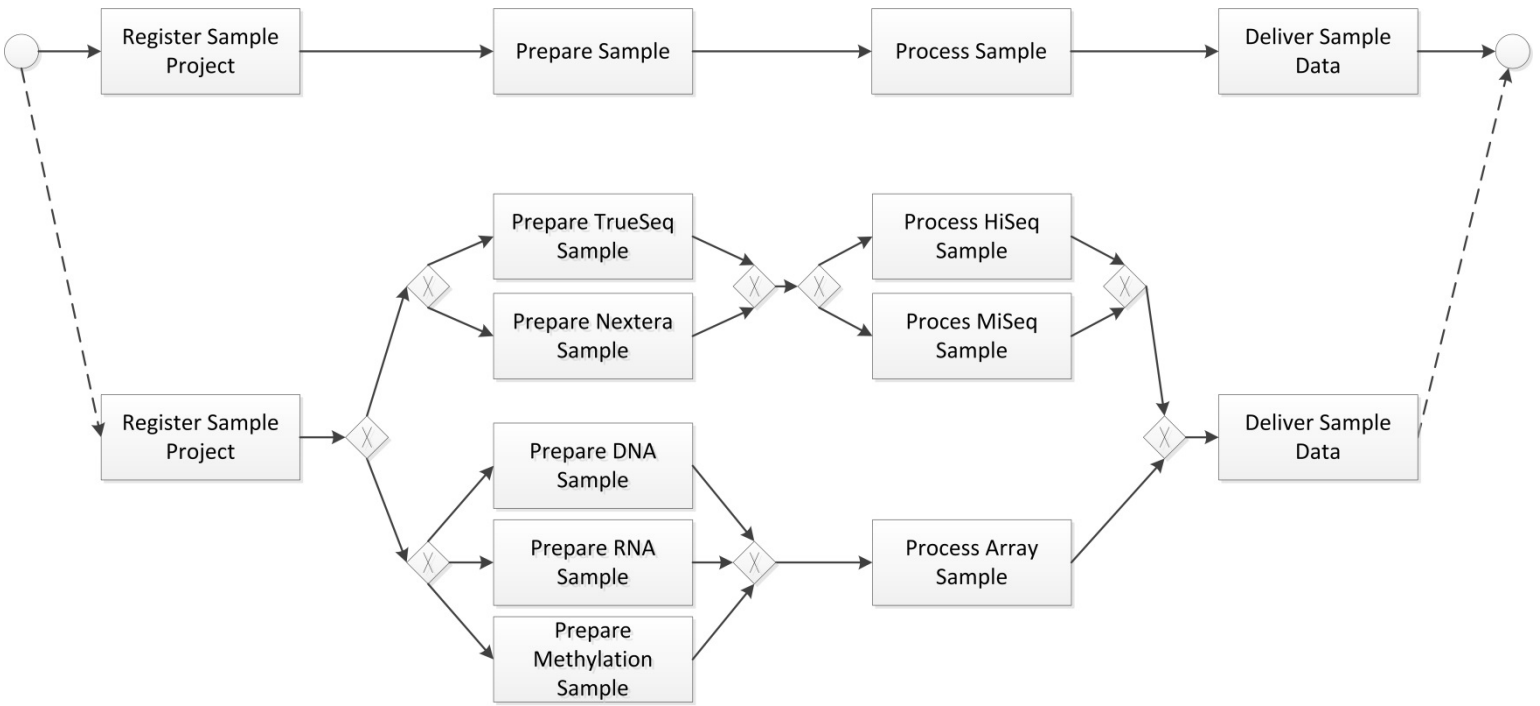


Figure 29: Variation map for DNA sequencing main process

6.5.2.7 Step 7b – Discovery of Process Models.

In this case study, there were no process models to begin with. There were, however "protocols" that explain on a very detailed procedural level, the steps to be performed for each specific case (variant). The lab has a little over 40 different "protocols" where the shortest is 20 pages and the longest is around 200 pages (about 4000 pages in total). These "protocols" also include processes that the core facility does not employ. On the other hand, the protocols cover only two of the sub-processes in the main process (Prepare Sample and Process Sample). The other sub-processes had not been previously documented.

For each sub-process of the main process, the process models were discovered and modeled in detail following the structure of variation map. For this work, the protocols, and information gathered from the domain experts (via hour-long weekly meetings for a period of 6 weeks) were used as input.

6.5.2.8 Step 8 – Model Data Object and Resource Variability

With the process models at hand, they were revisited for elicitation of data object and resource variability on the activity level. The procedure for this step was identical with that of the FXMM case study.

6.5.2.9 Step 9 – Verification of results by domain experts.

Once all processes had been modeled, a hand-over meeting was organized with the domain experts. Following this meeting, the models were examined in detail by the domain experts. As not all domain experts work with the same processes or have the same responsibilities, they divided the sub-processes in accordance with their area of expertise and responsibility. Thereafter, weekly meetings were held with the domain experts for 6 weeks. At each meeting, a subset of process models that had been examined, were verified. After the verification had been completed, the weekly meetings continued for the purpose of eliciting requirements from the process models for their future LIMS. As such, the process models were used as the primary source for eliciting requirements for their new information system. In fact, the first prototype of their information system was developed on the basis of the requirements elicited form the process models discovered using the decomposition driven method.

6.5.3 Execution of DNA Case Study using the Baseline Method

As mentioned previously, the same set of core facility business processes was modeled according to the approach outlined by Sharp and McDermott [167], so as to have a baseline for comparison. Sharp & McDermott's method (henceforth S&M) consists of the following steps:

Step 1: Identify the start events (called triggers) and end events (results) of the process.
Step 2: Identify major components based on milestones of the process (sub-processes).
Step 3: Identify the variants (cases) of the process.
Step 4: Identify internal and external stakeholders (participating organizations in the process).
Step 5: Identify for each participating organization, the individual actors their main responsibilities.
Step 6: Identify systems and data objects (supporting mechanisms) of the process.
Step 7: Conduct workshops where all tasks are listed and then sorted in order to create a flat process model.
Step 8: Identify the logical breakpoints of the flat process model and cluster activities within these points together as a sub-process.

The S&M method adopts a fragmented approach when modeling families of process variants. The method advocates for keeping variants in separate process models, arguing that design-time variation points should not be captured in a process model because these decisions have already been made prior to, and not during the execution of the process. However, if two variants are very similar, the method concedes that they can be modeled together, although this is not the preferred solution. Concretely, in case multiple variants have been identified, the method suggests to start by modeling one variant completely – for instance the most common one. This first variant is modeled flat. Next, the second variant (case) is taken and compared to the already modeled variant. If the variants are very similar, they can be modeled together. Note that the first five steps are conducted only once and step 7 and 8 are repeated for each additional variant. As such, for each additional variant that is different from the first process model, step 7 and 8 are repeated.

In order to minimize learning effects, the S&M method was applied in parallel with the method proposed in this thesis. The first two steps of S&M concern modeling the main process as part of the purpose of framing the project. Accordingly, while performing step 1 of the decomposition-driven method (modeling the main process), information needed for performing the first two steps of S&M was gathered. Similarly, step 2 (identify drivers) and step 4 (identify variants) of the decomposition-driven method correspond to steps 3 to 6 in S&M method, and thus these steps were done in parallel for both methods.

Steps 7 and 8 of the S&M method are the steps were the models are produced. Two researchers (one being the author of this thesis) began by modeling the most common variant, first as a flat process model, followed by the identification of logical breakpoints and extraction of sub-processes. Then, the next variant was taken and compared with the first already modeled variant. If they were very similar, they were modeled together. This procedure is repeated until all variants of the main process had been covered. Steps 7 and 8 of the S&M

method were performed in parallel with *Step 7b of the decomposition-driven method (see above)*.

It should be noted that the S&M do not provide guidance as how to manage sub-processes that are shared by several variants. In this case, refactoring was applied for this purpose, i.e. if several variants shared a sub-process, were modeled only once.

6.6 Findings

6.6.1 Findings from the Banking Case Study

As mentioned earlier, in the banking case study the original (input) process models had been modeled flat (no decomposition). In order to make them comparable with the models produced after consolidation, the flat process models were split into sub-processes following the same sub-process structure that resulted from the consolidation. In this way, the input process models and the consolidated ones are comparable in terms of hierarchical structure, although they differ in amount of duplication.

The input process models did not include NDF and bulk matching. These processes had only been partially documented in textual format prior to the consolidation. During the consolidation effort, these two processes were modeled as well. However, to make the input and the consolidated process models comparable, these were not taken into account in any of the statistics given below.

The input process models contain 35 sub-process models and 210 activity nodes (not counting gateways and artifacts such as data objects or data stores). Out of these, 75 activity nodes were duplicate occurrences (an activity occurring N times across all sub-process models counts as N duplicate occurrences). Thus, it can be said that the duplication rate in the input models is 36 %. Note that the 35 sub-process models in the input were distinct models, although some of them had strong similarities.

The consolidated models contain 17 sub-process models and 149 activity nodes of which 22 are duplicate occurrences, corresponding to 15 % duplication. Thus the consolidated models contain 30% less activity nodes, half of the sub-process models and half of the duplication rate relative to the original model. These observations (summarized in Table 22) support the hypothesis of the case study.

Table 22: Size Metrics before and after Process Model Consolidation

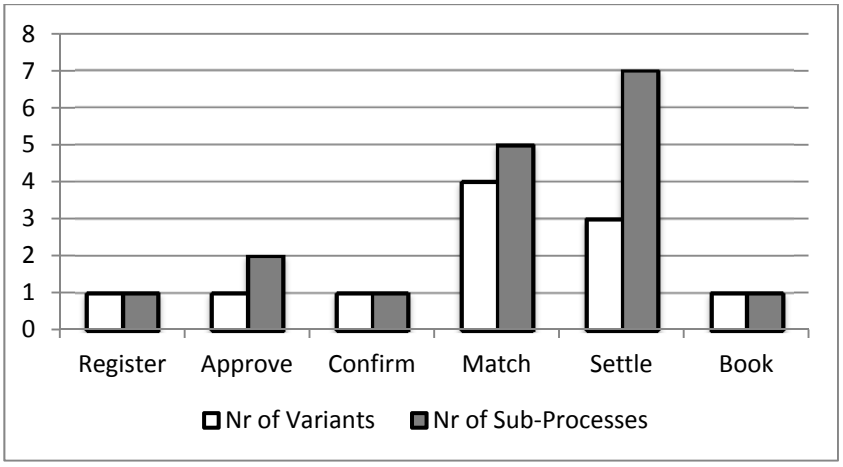
Variable	Input	Consolidated
Main Process Models	4	1
Sub-Process Models	35	17
Activity Nodes	210	149
Duplicate Activity Occurrences	75	22
Duplication rate	36 %	15 %
CNC	1,25	1,33

One can expect that the complexity of the process models will increase during consolidation since additional gateways are introduced to capture differences between multiple variants of a sub-process model. The consolidation of process models naturally affected their complexity. For instance, there were four separate sections of the flat process models corresponding to "register trade". The input process model for corporate clients (trading both FX and MM) was the least complex one with a CNC of 0.8. For interbank trading of FX (both gross and CLS), the corresponding CNC was 1.09. For the interbank trading of MM, it was 1.17. The combined complexity of the input "register" process models were 1.07. The consolidated sub-process (only one as the variants were similar and lacking strong variation driver), have a CNC of 1.11. This trade-off between reduction in duplication and increase in complexity has also been observed in [150].

The input process models had four main processes, one for corporate clients trading both FX and MM, two for interbank trading of FX via gross or CLS and finally one for interbank trading of MM (both gross and CLS). Therefore, there was no distinct driver behind the segregation of the process models. In one case it was based on customer type (corporate versus interbank) regardless of product. In another case it was based on product (FX versus MM) and a third one was based on how the trades were settled (gross or CLS). In contrast to this, the consolidated models had one common main process, where the business drivers for variations are expressed at each sub-process of the main process. For instance, for "confirm trade" the driver is based on product (FX/MM versus NDF), and for "match trade" it is based on customer (corporate versus interbank). As such, the consolidated set of process models were also a restructuring of how the business process is captured. The structure of the main process changed from four flat input main processes to one that encompasses all four by expressing its variability as depicted in the variation map (cf. Fig. 26). Furthermore, the variability (as expressed in number of variants) and number of sub-processes are as most intensive in the middle section of the main process as can

be seen in Table 23. For “approve”, “match”, and “settle”, there are three levels of decomposition.

Table 23: Number of Variants and Sub-processes in the Main Process



The results from data object and resource variability show that there are occurrences of such variability but their number is not of such magnitude as to make an impact on the process models. In the 17 sub-processes, a total of 8 process models had such variability on activity level. In total, 10 data object driven occurrences of variability were identified and one resource induced variability. However, only one of these resulted in modification of how the activity was modeled. As such, only one process model out of 17 were affected (a re-modeling of an activity).

6.6.2 Findings of the DNA Sequencing Case Study

In the DNA case study, the same set of processes was modeled using the decomposition-driven method and the S&M method as a baseline. The baseline method led to 110 process models, comprising 379 activity nodes (only counting sub-processes and tasks). The duplicity count is 218 (as defined in Section 6.1). Thus the process models in the baseline method had a duplication rate of 41%. On the other hand, the process variants modeled according to the method of this thesis had 379 activity nodes with a duplicity count of 92 (i.e. 20 %). Compared to the baseline, this method resulted in a reduction of duplication of about 50%. Furthermore, this method led to 83 sub-process models whereas the baseline required 110, i.e. 33% more. These observations (summarized in Table 24) support the hypothesis that by applying the decomposition-driven method, a

family of process variants can be represented using fewer activities and sub-process models compared to a fragmented approach.

Table 24: Size Metrics for Decomposition-Driven Method versus Baseline Scenario

Variable	**S&M**	**Decomposition-driven method**
Process Models	110	83
Sub-Processes	147	93
Activity Nodes (Tasks)	379	371
Duplicate Activity Occurrences	218	92
Duplication rate	41%	20 %
CNC	0,9	0,97

Similar to the first case study, CNC metric was used to compare the output of the decomposition-drive method with the baseline. In the baseline scenario, each variant of for instance "prepare sample" was modeled separately. This resulted in a total of 7 sub-processes, one for each variant of "prepare trueseq sample". These models have a CNC of between 0.88 and 0.91. However, as they were similar and lacked strong variation driver, they have been modeled together. As such, the complexity of this sub-process is higher as it encompasses a total of 7 variants and has a CNC of 1.24. It can be noted that, as in the previous case, a trade-off between number of process models and complexity. The baseline process models have a total of 739 arcs and 822 nodes (sub-processes, tasks, gateways and events), which gives an average CNC of circa 0.9.[1] On the other hand, the set of process models obtained via the decomposition-method have 753 arcs and 773 nodes, thus an average CNC of around 0.97. This marginally higher CNC value should be contrasted with the significant reduction in the number of process models (110 versus 83) and duplication rate (41% versus 20%).

Similar to the FX and MM case, it can be seen that the variability in the process occurs more towards the mid section of the main process. In this case, there are four sub-processes of the main process as can be seen in Fig. 29. The first sub-process, "register sample project" does not have any variability on the level of the variation map. However, at lower levels of decomposition, there is variability in terms of method used to measure the quality of the samples. In "prepare sample" there are five variants at the level of the variation map and in total,

[1] This relatively low CNC value stems from the fact that a majority of processes are derived from laboratory protocols that are highly sequential (i.e. relatively few branching points).

variants and finally, in “deliver sample data” there is only one variant. In total, the DNA case has 27 variants. This is also reflected in the number of sub-processes under each main sub-process of the main process. In “register sample project” there are a total of 6 sub-processes. The numbers for “prepare sample”, “process sample” and “deliver sample” are 64, 22 and 1 respectively. The DNA case has more levels of decomposition (up to five levels) as compared to the baseline scenario (3 levels). As such, the baseline scenario has fewer levels of process model hierarchy (more flat) but is larger whereas the models discovered through decomposition-driven method, have more levels of hierarchy (more deep) but are overall smaller.

The results from data object and resources induced variability in the process models show that out of the 83 process models, activities in 14 process models had variability due to data objects or resources. In total, 28 data object and one resource induced variability of activities were identified. Of these 29 nine cases, only 6 resulted in re-modeling. As such, about 8% of the activities had variability induced by data objects or resources and of these, only 20% resulted in re-modeling. In other words, less than 2% of all activities were re-modeled due to data object or resource induced variability. Clearly, data objects and resources, as a source of variability on activity level, have a very limited affect on the modeling of the processes.

6.7 Threats to Validity

When conducting case studies, there are threats to validity that ought to be considered, particularly regarding construct validity, external validity and reliability [156]. In order to reduce the general threats to validity, the following measures were taken.

Triangulation [55, 157]: Data was collected from different sources to increase the validity of the study. The main sources were interviews and artifacts, both describing the same business processes. Triangulation should improve construct validity, as several sources of data will identify potential discrepancies between what is studied, as perceived by the researcher, and what the interviewed persons have understood to be the object of the study. Triangulation will also reduce the risk of internal validity because more sources of data representing the same business processes will show different aspects of the processes under study and therefore reduce the risk of the researcher being unaware of them. Furthermore, triangulation increases the reliability of the study, as with more data representing the same set of business processes, there is less “room” for biased interpretation.

These case studies presented here seek to describe how work is being conducted rather than understanding why the work is performed as it is. As such, interpretation of data is not imperative for the results.

Peer debriefing [55, 157]: Another measure taken to reduce the threat to validity is peer debriefing. Two additional researchers reviewed the case study

designs. In addition, the results were continuously be presented to a group of researchers, all highly familiar with business process management, in order to reduce bias, identify potential improvements, review the results and ensure correctness in the research process. These measures were taken to improve construct validity, help strengthen internal validity and increase reliability of the study.

Member checking [55, 157]: The results of all interviews and workshops, as well as the final process models were checked and validated by the participants in the case studies. They reviewed the material in detail in order to ensure that the process models were correct. This improved the reliability of the study as it reduces the risk of faults in the "transcripts" (from meetings and workshops).

6.7.1 External Validity

External validity concerns the extent to which the findings can be generalized beyond the setting of the study. The decomposition driven method has been applied on two case studies, and in line with the inherent limitation of case study methods, the results are limited in the extent they can be generalized. The results are naturally dependent on the domain experts and the purpose of the study, and it can, therefore, limit the repetitiveness of the decomposition driven method. Hence the method is replicable but results may vary due to aforementioned reasons. In addition, the FXMM case was perhaps easier to manage due to relatively less number of process models and in the DNA case was not of high degree of complexity (in terms of having mostly sequential processes). As such, the application of the method has not been tried and must be seen as a limitation on the generalizing of the results for the time being. On the other hand, it should be underscored that the case studies have been conducted in two different industrial settings and contexts with active involvement of domain experts with very different backgrounds.

6.7.2 Reliability

Reliability refers to the level of dependency between the results and the researcher, i.e. would the same results be produced if another researcher conducted the study. This threat was to some extent tackled by having several series of verifications by the domain experts without the presence of the researcher (member checking as defined in [157]). Furthermore, as described above, peer debriefing [157] was applied to further ensure better reliability. In addition, by applying both data triangulation (using both protocols and domain experts) and observer triangulation (two authors of this paper involved), reduced the threat to validity [157].

6.7.3 Construct Validity

Construct validity refers to the extent to which the tools used, measure what the researcher has in mind and also what is being investigated. This risk was reduced as the results of the case studies, were de facto used. As mentioned before, the purpose of the consolidated process models for the bank case study was to use them in evaluating off-the-shelf products. The consolidated process models were used in a four-day workshop with a supplier of off-the-shelf solution to investigate the extent to which the solution could satisfy their needs. The consolidated process models were key to the whole evaluation process. The workshop structure, functionality covered, test cases and evaluation criteria were all based on the consolidated process models.

The discovered process models for the core facility case study were used to understand their processes, to elicit requirements for a comprehensive LIMS system and used to develop a prototype LIMS system.

7 CONCLUSION

When a business process needs to be modeled, be it for educational purposes or for eliciting requirements for the development of a new information system, it is not a trivial thing to determine how to model the processes. When the business process has variants, the complexity increases. An easy approach would be to model each variant separately as an independent process model. Such an approach would cause large number of process models with high degree of duplicity. This would counteract the purpose of the models, i.e., to be easily understandable when working with them and keeping them up-to-date. On the other hand, all variants could be modeled as one large process model but such an effort would also counteract the purpose of the models.

In light of this, how can a family of process variants be modeled in a way that (1) reduces the number of process models and duplicity, (2) does not become overly complex and (3) is aligned with the business processes it aims at representing. As such, the overall research question of this thesis is "*how can a family of process variants be modeled when consolidating or discovering business process models?*"

Variability in process models being consolidated or discovered can quickly become very complex. Such complexity is commonly managed by decomposition of process models into more manageable parts. As such, it is necessary to address the question of "*how can process models be decomposed*?"

Decomposition is made vertically (slicing) and horizontally (dicing). As to vertical decomposition, this thesis proposes a variant-driven decomposition of process models based on the business driver (the root cause) and similarity of the process variants. As such, a sub-processes is "sliced" by considering the underlying business reason for the existance of variants and the degree to which the variants are similar to each other.

In this thesis, it is shown that decomposition can be based on several heuristics as presented in chapter 3. Some of these heuristics (such as repetition, shared processes and role based) are not generally applicable on process models, as they require certain conditions to be fulfilled. However heuristics based on breakpoint and data object provide necessary criteria for decomposition but the determination of which fragments represent a breakpoint in the process or share the same set of data objects is somewhat unclear and dependent on subjective assessment. In conclusion, breakpoint and data object (if data objects are available and modeled in a consistent manner in the process models) are the closest to being complete and therefore better suited to be used for dicing a process model. It should be noted that domain experts are most likely better acquainted with the flow of their processes rather than the objects and as such, they might prefer breakpoint heuristics.

Based on these foundations, the question of "*how can a family of process variants be modeled"* is addressed by the development of a decomposition driven method for consolidating or discovering business process models. The method proposed address the research question with an integrated divide-and-

conquer method for modeling business processes with variants. As such, the method proposes to begin with the highest level of the process (the main process or the value chain) and then, gradually postponing which variants to model and which parts of variants to model in more detail.

In this context, the following propositions were investigated: (1) Decisions to model multiple process variants separately should be taken at the lowest possible level of process decomposition rather than upfront. In other words, rather than deciding upfront to split multiple process variants into separate models, one should consider postponing this decision to the level of each sub-process, until there are strong reasons for modeling the variants separately. (2) Decisions on whether to model variants together or separately should be based on the business drivers for variation (the extent to which the separation between variants are integral to the business) and syntactic drivers (the degree of similarity between variants).

The method is validated by applying it to two case studies: one aimed at consolidating an existing collection of models, and the other aimed at modeling a family of process variants from scratch. Although not fully generalizable, the case study findings show that the proposed method provides a structured approach to modeling families of process variants in a way that reduces duplication with relatively minor penalty on model complexity and is aligned with the business processes it represents.

The method has been formulated in an intra-enterprise setting where all the stakeholders are able to agree on the primary drivers of variability in the business processes. When applying the method in a cross-organizational process, additional issues might arise. For example, two business partners might have different viewpoints of the relative strengths of the drivers. This situation would require a compromise, which is not considered in this method.

Furthermore, the method has been developed in the context of a procedural process modeling language (e.g. BPMN) where decision points are explicitly represented along the flow of activities. Recently, alternative process modeling paradigms based on declarative styles have been proposed [7]. In declarative process models, activity flows and decision points are neither exhaustively nor explicitly captured and hence the proposed method is not directly applicable. Extending the method to cater for cross-organizational processes and declarative process modeling are avenues for future work.

8 REFERENCES

1. Aalst W vad Der, Weijters AJMM (2004) Process mining: a research agenda. Comput Ind 53:231–244.
2. Aalst W Van Der (1999) Generic Workflow Models: How to Handle Dynamic Change and Capture Management Information. Proc. Fourth IFCIS Int. Conf. Coop. Inf. Syst. IEEE, pp 115–126
3. Aalst W Van Der (2011) Business process configuration in the cloud: How to support and analyze multi-tenant processes? Proc. – 9th IEEE Eur. Conf. Web Serv. ECOWS 2011. pp 3–10
4. Aalst W Van Der, Dreiling A, Gottschalk F, Rosemann M, Jansen-Vullers MH (2006) Configurable Process Models as a Basis for Reference Modeling. Bus. Process Manag. Work. Lect. Notes Comput. Sci. Springer, pp 512–518
5. Aalst W Van Der, Dumas M, Gottschalk F, Ter Hofstede AHM, La Rosa M, Mendling J (2010) Preserving correctness during business process model configuration. Form Asp Comput 22:459–482.
6. Aalst W Van Der, Jablonski S (2000) Dealing with workflow change: Identification of issues and solutions. Comput Syst Sci Eng 15:267–276.
7. Aalst W Van Der, Pesic M, Schonenberg H (2009) Declarative workflows: Balancing between flexibility and support. Comput Sci - Res Dev 23:99–113.
8. Aalst W Van Der, Rosemann M, Dumas M (2007) Deadline-based escalation in process-aware information systems. Decis Support Syst 43:492–511.
9. Aalst W Van Der, Weijters T, Maruster L (2004) Workflow mining: Discovering process models from event logs. IEEE Trans Knowl Data Eng 16:1128–1142.
10. Acher M, Collet P, Lahire P, France R (2010) Managing variability in workflow with feature model composition operators. Lect. Notes Comput. Sci. Springer, pp 17–33
11. Agrawal R, Gunopulos D, Leymann F (1998) Mining Process Models from Workow Logs. 6th Int. Conf. Extending Database Technol. Springer Berlin Heidelberg, pp 467–483
12. Altuhhova O, Matulevičius R, Ahmed N (2012) Towards definition of secure business processes. Lect Notes Bus Inf Process 112 LNBIP:1–15.
13. Ang Z, Massingham P (2007) National culture and the standardization versus adaptation of knowledge management. J Knowl Manag 11:5–21.
14. Antón A, McCracken W, Potts C (1994) Goal decomposition and scenario analysis in business process reengineering. J Adv Inf Syst 811: 94–104.
15. Atkinson C, Bayer J, Muthig D (2000) Component-based product line development: the KobrA approach. Kluwer Int. Ser. Eng. Comput. Sci. Springer, pp 289–310

16. Atkinson C, Bostan P, Brenner D, Falcone G, Gutheil M, Hummel O, Juhasz M, Stoll D (2008) Modeling components and component-based systems in KobrA. Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics). Springer, pp 54–84
17. Bailey K (1994) Typologies and taxonomies: an introduction to classification techniques. SAGE Publications, Inc., Thousands Oaks, California
18. Batory D, Geraci B (1997) Composition Validation and Subjectivity in Gen Voca Generators. IEEE Trans Softw Eng 67–82.
19. Becker J, Becker J, Winkelmann A (2009) Developing a Business Process Modeling Language for the Banking Sector – A Design Science Approach Developing a Business Process Modeling Language for the Banking Sector – A Design Science Approach. AMCIS 2009 Proc.
20. Becker J, Delfmann P, Dreiling A, Knackstedt R, Kuropka D (2004) Configurative Process Modeling–Outlining an Approach to increased Business Process Model Usability. Proc. 2004 Inf. Resour. Manag. Assoc. Conf. pp 615–619
21. Becker J, Delfmann P, Knackstedt R (2007) Adaptive Reference Modeling: Integrative Configurative and Generic Adaption Techniques for Information Models. Ref. Model. Springer, pp 27–58
22. Benbasat I (1987) The Case Research Strategy in Studies of Information Systems Case Research. MIS Q 3:369–386.
23. Berger IR, Soffer P, Sturm A (2009) Organisational reference models: supporting an adequate design of local business processes. Int J Bus Process Integr Manag 4:134.
24. Braunnagel D, Johannsen F, Leist S (2014) Coupling and process modeling – An analysis at hand of the eEPC. Modellierung. pp 121–136
25. Brocke J vom, Rosemann M (2010) Handbook on Business Process Management Vol. 1. Springer, Heidelberg
26. Buijs, J.C.A.M., Dongen, B.F. and Aalst WMP (2012) Towards Cross-Organizational ProcessMining in Collections of ProcessModels and Their Executions. Bus. Process Manag. Work. Lect. Notes Bus. Inf. Process. pp 2–13
27. Burton-Jones A, Meso PN (2004) Conceptualizing Systems for Understanding : An Empirical Test of Decomposition Principles in Object Oriented Analysis Working Paper Conceptualizing Systems for Understanding : An Empirical Test of Decomposition Principles in Object Oriented Analysis. Inf Syst 17:38–60.
28. Cardoso J (2005) How to Measure the Control-flow Complexity of Web Process and Workflows. Work. Handb. 2005. pp 199–212
29. Cardoso J, Mendling J (2006) A discourse on complexity of process models. Bus. Process Manag. Worshops, Lect. Notes Comput. Sci. Springer, pp 117–128

30. Casati F, Ilnicki S, Jin L, Krishnamoorthy V, Shan M-C (2000) Adaptive and Dynamic Service Composition in eFlow. Adv Inf Syst Eng 1789: 13–31.
31. Churcher NI, Shepperd MJ, Chidamber S, Kemerer CF (1995) Comments on "A metrics suite for object oriented design. IEEE Trans Softw Eng 21:263–265.
32. Ciuksys D, Caplinskas a (2007) Reusing ontological knowledge about business processes in IS engineering: process configuration problem. Informatica 18:585–602.
33. Cohen SG, Hess JA, Novak WE, Peterson a S (1990) Feasibility Study Feature-Oriented Domain Analysis (FODA). DTIC Document
34. Conforti R, Dumas M, García-Bañuelos L, La Rosa M (2014) Beyond Tasks and Gateways: Discovering BPMN Models with Subprocesses, Boundary Events and Activity Markers. Bus. Process Manag. Lect. Notes Comput. Sci. Springer International Publishing, pp 101–117
35. Daneva M, Heib R, Scheer A (1996) Benchmarking business process models. Technical Report, Saarland University
36. Davenport TH (1993) Selecting Processes for Innovation. Process Innov. reengineering Work through Inf. Technol. Harvard University Press, pp 27–36
37. Davenport TH, Davenport TH, Short JE, Short JE (1990) The new industrial Engineering: Information Technology and Business Process Redesign. Sloan Manage Rev 31:11–27.
38. Davenport TH, Short JE (1990) The New Industrial Engineering: Information Technology and Business Process Redesign. Management 31:46.
39. Davis R (2001) Business Process Modelling With Aris: A Practical Guide. Springer Science & Business Media
40. Davis R, Brabander E (2008) ARIS Design Platform. Springer Science & Business Media
41. Delfmann P (2006) Adaptive Referenzmodellierung. Methodische Konzepte zur Konstruktion und Anwendung wiederverwendungsorientierter Informationsmodelle. Adv Inf Syst Manag Sci 25:266.
42. Dibb S, Stern P, Wensley R (2002) Marketing knowledge and the value of segmentation. Mark. Intell. Plan.
43. Dijkman R, Dumas M, Van Dongen B, Krik R, Mendling J (2011) Similarity of business process models: Metrics and evaluation. Inf Syst 36:498–516.
44. Dijkman R, Dumas M, García-Bañuelos L (2009) Graph matching algorithms for business process model similarity search. Bus Process Manag Business P:48–63.
45. Dijkman R, Dumas M, Garcia-Banuelos L, Kaarik R (2009) Aligning Business Process Models. 2009 IEEE Int. Enterp. Distrib. Object Comput. Conf.

46. Dijkman R, Gfeller B, Küster J, Völzer H (2011) Identifying refactoring opportunities in process model repositories. Inf Softw Technol 53: 937–948.
47. Dijkman R, Grefen P, Weele a van (2012) A literature review in process harmonization: a conceptual framework.
48. Dijkman R, Rosa M La, Reijers H a. (2012) Managing large collections of business process models—Current techniques and challenges. Comput Ind 63:91–97.
49. Dijkman R, Vanderfeesten I, Reijers H (2011) The Road to a Business Process Architecture: An Overview of Approaches and their Use. Einhoven University of Technology
50. Dijkman RM, Dumas M, Ouyang C (2008) Semantics and analysis of business process models in BPMN. Inf Softw Technol 50:1281–1294.
51. Dongen B van, Dijkman R, Mendling J (2008) Measuring similarity between business process models. Lect Notes Comput Sci (including Subser Lect Notes Artif Intell Lect Notes Bioinformatics) 5074 LNCS:450–464.
52. Dumas M (2012) Consolidated management of business process variants. Lect. Notes Bus. Inf. Process. Springer Berlin Heidelberg, p 1
53. Dumas M, García-Bañuelos L, Dijkman RM (2010) Similarity Search of Business Process Models. IEEE Data Eng Bull 32:23–28.
54. Dumas M, La Rosa M, Mendling J, Reijers H (2013) Fundamentals of business process management. Springer Science & Business Media, Heidelberg
55. Easterbrook S, Singer J, Storey M-A, Damian D (2008) Selecting Empirical Methods for Software Engineering Research. Guid to Adv Empir Softw Eng 285–311.
56. Eberle H, Unger T, Leymann F (2009) Process fragments. Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics). pp 398–405
57. Eisenecker U, Czarnecki K (2000) Generative Programming: Methods, Tools, and Applications. Springer Science & Business Media
58. Ekanayake CC, Dumas M, García-Bañuelos L, La Rosa M (2013) Slice, mine and dice: Complexity-aware automated discovery of business process models. Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics). pp 49–64
59. Eppinger SD, Whintey DE, Smith RP, Gebala D a (1994) A Model-Based Method for Organizing Task in Product Development. Res Eng Des 6:1–13.
60. Flyvbjerg B (2006) Five Misunderstandings About Case-Study Research. Qual Inq 12:219–245.
61. Foedermayr EK, Diamantopoulos A (2008) Market Segmentation in Practice: Review of Empirical Studies, Methodological Assessment, and Agenda for Future Research. J Strateg Mark 16:223–265.
62. Gerring J (2004) What is a case study and what is it good for? Am Polit Sci Rev 98:341–354.

63. Gerth C, Engels G (2008) Detecting and Resolving Process Model Differences in. Differences 244–260.
64. Gerth C, Küster JM, Luckey M, Engels G (2010) Precise detection of conflicting change operations using process model terms. Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics). Springer, pp 93–107
65. Ghose A, Koliadis G, Chueng A (2007) Process discovery from model and text artefacts. Proc. - 2007 IEEE Congr. Serv. Serv. 2007. pp 167–174
66. Girod S, Bellin JB (2011) Revisiting the "Modern" Multinational Enterprise Theory: An Emerging-market Multinational Perspective. Futur Foreign Direct Invest Multinatl Enterp (Research Glob Strateg Manag Vol 15) 167–210.
67. Gottschalk F, van der Aalst WMP, Jansen-Vullers M (2007) Configurable Process Models - A Foundational Approach. Ref Model 59–77.
68. Gottschalk F, Van Der Aalst WMP, Jansen-Vullers MH (2008) Merging event-driven process chains. Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics). Springer, pp 418–426
69. Hahn C, Zinnikus I (2008) Modeling and executing service interactions using an agent-oriented modeling language. CEUR Workshop Proc 344:37–40.
70. Hall JM, Johnson ME (2009) When Should a Process be Art,Not Science. Harv Bus Rev 87:58–65.
71. Hallerbach a, Bauer T, Reichert M (2008) Managing process variants in the process life cycle. ICEIS 2008 - Proc 10th Int Conf Enterp Inf Syst 2 ISAS:154–161.
72. Hallerbach A, Bauer T, Reichert M (2010) Configuration and management of process variants. In: Brocke J, Rosemann M (eds) Handb. Bus. Process Manag. 1. Springer Berlin Heidelberg, pp 237–255
73. Halmans G, Pohl K (2004) Communicating the variability of a software-product family to customers. Inform – Forsch und Entwicklung 18:113–131.
74. Hammer M (1990) Reengineering work: don't automate, obliterate. Harv Bus Rev 68:104–112.
75. Hammer M (2010) What is Business Process Management? Handb. Bus. Process Manag. 1 – Introd. Methods, Inf. Syst. pp 3–16
76. Hammer M, Champy J (1994) Reengineering the Corporation – A Manifesto For Business Revolution. Acad Manag Rev 19:595.
77. Harmon P (2003) Business Process Change: A Manager's Guide to Improving, Redesigning, and Automating Processes. Morgan Kaufmann, San Francisco
78. Ivanović D, Carro M, Hermenegildo M (2010) Automatic fragment identification in workflows based on sharing analysis. Lect Notes Comput Sci (including Subser Lect Notes Artif Intell Lect Notes Bioinformatics) 6470 LNCS:350–364.
79. James PTJ (1996) Total quality management : an introductory text. Prentice Hall, New York

80. Joeris G, Herzog O (1998) Managing evolving workflow specifications. Proceedings. 3rd IFCIS Int. Conf. Coop. Inf. Syst. (Cat. No.98EX122). IEEE, pp 310–319
81. Johannsen F, Leist S (2012) Wand and Weber's Decomposition Model in the Context of Business Process Modeling. Bus Inf Syst Eng 4:275–286.
82. Jr NFK (1996) Product flow, breadth and complexity of business processes: An empirical study of 15 business processes in three organizations. Bus Process Re-engineering & Manag J 2:8–22.
83. Jung J-Y, Bae J, Gavrilova M, Gervasi O, Kumar V, Tan C, Taniar D, Laganá A, Mun Y, Choo H (2006) Workflow Clustering Method Based on Process. Comput. Sci. Its Appl. – ICCSA 2006 Int. Conf. Glas. UK, May 8–11, 2006. Proceedings, Part II. Springer, pp 379–389
84. Kakaomerlioglu DC, Carlsson B (1999) Manufacturing In Decline? A Matter Of Definition. Econ Innov New Technol 8:175–196.
85. Kettinger W, Teng J, Guha S (1997) Business process change: a study of methodologies, techniques, and tools. MIS Q 21:55–80.
86. Khalaf R, Leymann F (2006) E Role-based Ddecomposition of Business Processes using BPEL. Proceeding ICWS '06 Proc IEEE Int Conf Web Serv 770–780.
87. Kim K, Won J, Kim C (2005) A fragment-driven process modeling methodology. Sci. Its Appl. 2005
88. Kitchenham B (2004) Procedures for performing systematic reviews. Keele, UK, Keele Univ 33:28.
89. Kitchenham B, Pickard L, Pfleeger SL (1995) Case studies for method and tool evaluation. IEEE Softw 12:52–62. doi: 10.1109/52.391832
90. Kleijn MJ (1998) An overview of inventory systems with several demand classes 1 Introduction 2 Examples of multiple demand classes. 1–13.
91. Korherr B, List B (2007) A UML 2 profile for variability models and their dependency to business processes. Proc. – Int. Work. Database Expert Syst. Appl. DEXA. IEEE, pp 829–834
92. Kristof W (1964) an Empirical Investigation on the Classification of Feelings. Psychol Forsch 28:46–63.
93. Kueng P, Kawalek P (1997) Goal-based business process models: creation and evaluation. Bus Process Manag J 3:17–38.
94. Kulkarni V, Barat S (2011) Business process families using model-driven techniques. Lect. Notes Bus. Inf. Process. Springer, pp 314–325
95. Kumar A, Yao W (2009) Process materialization using templates and rules to design flexible process models. Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics). Springer, pp 122–136
96. Kumar A, Yao W (2012) Design and management of flexible process variants using templates and rules. Comput Ind 63:112–130.
97. Kusiak a., Wang J (1993) Efficient organizing of design activities. Int J Prod Res 31:753–769.
98. Lambert S (2006) A Business Model Research Schema. BLED Proc 1–13.

99. Lamsweerde A Van (2001) Goal-Oriented Requirements Engineering: A Guided Tour. Proc. Fifth IEEE Int. Symp. Requir. Eng. IEEE Comput. Soc. pp 249–262
100. Latva-Koivisto A (2001) Finding a complexity measure for business process models. Technical Report, Technological Systems Analysis, Helsinki University. pp 1–26
101. Li C, Reichert M, Wombacher A (2009) A Heuristic Approach for Discovering Reference Models by Mining Process Model Variants. Organization 1–51.
102. Li C, Reichert M, Wombacher A (2010) The Minadept Clustering Approach for Discovering Reference Process Models Out of Process Variants. Int J Coop Inf Syst 19:159–203.
103. List B, Korherr B (2006) An Evaluation of Conceptual Business Process Modelling Languages. 2006 ACM Symp Appl Comput 1532–1539.
104. Ludwig H, Rankin Y, Enyedi R, Anderson LC (2011) Process variation analysis using empirical methods: a case study. Proc. Bus. Process Manag. Springer, pp 62–65
105. Malone TW, Crowston K, Lee JLJ, Pentland B (1993) Tools for inventing organizations: toward a handbook of organizational processes. [1993] Proc Second Work Enabling Technol Collab Enterp 45:425–443.
106. Manez J A. (2001) Multiproduct firms and product differentiation: a survey. Economics Department, Univsity of Warwick UK. Nr. 594
107. Mannion M, Mannion M (2002) Using First-Order Logic for Product Line Model Validation. Softw Prod Lines, Second Int Conf 2379:176–187.
108. Manrodt KB, Vitasek K (2004) Global Process Standardization: A Case Study. J Bus Logist 25:1–24.
109. Mansar SL, Reijers H a. (2005) Best practices in business process redesign: Validation of a redesign framework. Comput Ind 56:457–471.
110. Da Matta Vegi LF, Peixoto DA, Soares LS, Lisboa-Filho J, De Paiva Oliveira A (2012) Business Process Management Workshops. Lect. Notes Bus. Inf. Process. Springer Berlin Heidelberg, pp 338–343
111. McKelvey B (1982) Organizational systematics-taxonomy, evolution, classification. University of California Press
112. Mendling J (2006) Testing Density as a Complexity Metric for EPCs. Techical Report, Vienna University of Economics and Business Administration, Wien
113. Mendling J, Neumann G, Van Der Aalst W, Aalst W Van Der (2007) Understanding the occurrence of errors in process models based on metrics. Lect. Notes Comput. Sci. pp 113–130
114. Mendling J, Reijers H a., van der Aalst WMP (2010) Seven process modeling guidelines (7PMG). Inf Softw Technol 52:127–136.
115. Mendling J, Simon C (2006) Business Process Design by View Integration. Bus. Process Manag. Work. Springer, pp 55–64
116. Milani F, Dumas M, Matulevičius R (2013) Decomposition driven consolidation of process models. Lect. Notes Comput. Sci. (including

Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics). Springer Berlin Heidelberg, pp 193–207
117. Moon M, Hong M, Yeom K (2008) Two-level variability analysis for business process with reusability and extensibility. Proc. – Int. Comput. Softw. Appl. Conf. pp 263–270
118. Muehlen M Zur, Wisnosky D, Kindrick J (2010) Primitives: design guidelines and architecture for BPMN models. Australas. Conf. Inf. Syst.
119. Muketha G, Ghani A (2010) A survey of Business Processes Complexity Metrics – 2010.PDF. Inf Technol J 9:1336–1344.
120. Nguyen T, Colman A, Han J (2011) Modeling and managing variability in process-based service compositions. Lect Notes Comput Sci (including Subser Lect Notes Artif Intell Lect Notes Bioinformatics) 7084 LNCS: 404–420.
121. Ould M (1997) Designing a re-engineering proof process architecture. Bus Process Manag J 3:232–247.
122. Pascalau E, Rath C (2010) Managing business process variants at eBay. Lect. Notes Bus. Inf. Process. Springer Berlin Heidelberg, pp 91–105
123. Paulson D, Wand Y (1992) An automated approach to information systems decomposition. IEEE Trans Softw Eng 18:174–189.
124. Perry D, Sim SE, Easterbrook S (2004) Case Studies for Software Engineers. IEEE, ICSE 2004, Proc. 26th Int. Conf. Softw. Eng. pp 736–738
125. Pimmler TU, Eppinger SD (1994) Integration analysis of product decompositions. Alfred P. Sloan School of Management, MIT, Cambridge, MA
126. Pohl K (2010) Requirements Engineering: Fundamentals, Principles, and Techniques. Springer Publishing Company, Incorporated
127. Pohl, Klaus, Böckle, Günter, van der Linden F (2005) Software Product Line Engineering – Foundations, Principles, and Techniques.
128. Polyvyanyy A, Smirnov S, Weske M (2008) Process model abstraction: A slider approach. Proc. – 12th IEEE Int. Enterp. Distrib. Object Comput. Conf. EDOC 2008. IEEE, pp 325–331
129. Polyvyanyy A, Smirnov S, Weske M (2009) The triconnected abstraction of process models. In: Dayal U, Eder J, Koehler J, Reijers H (eds) Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics). Springer Berlin Heidelberg, pp 229–244
130. Polyvyanyy A, Smirnov S, Weske M (2010) Business Process Model Abstraction. Handb. Bus. Process Manag. 1. Springer, pp 149–166
131. Porter M (2008) Competitive advantage: Creating and sustaining superior performance. Simon and Schuster, New York
132. Puhlmann F, Schnieders A, Weiland J, Weske M (2005) Variability mechanisms for process models. PESOA-Report TR 17:10–61.
133. Qiao M, Akkiraju R, Rembert AJ (2011) Towards efficient business process clustering and retrieval: Combining language modeling and structure matching. Lect Notes Comput Sci (including Subser Lect Notes Artif Intell Lect Notes Bioinformatics) 6896 LNCS:199–214.

134. Razavian M, Khosravi R (2008) Modeling Variability in Business Process Models Using UML. Fifth Int. Conf. Inf. Technol. New Gener. (itng 2008)
135. Reichert M, Weber B (2012) Enabling Flexibility in Process-Aware Information Systems: challenges, methods, technologies. Springer Science & Business Media, Heidelberg
136. Reijers H a. (2003) A Cohesion Metric for the Definition of Activities in a Workflow Process. Proc. EMMSAD. pp 116–125
137. Reijers H a., Mans RS, van der Toorn R a. (2009) Improved model management with aggregated business process models. Data Knowl Eng 68:221–243.
138. Reijers H a., Mendling J, Dijkman RM (2011) Human and automatic modularizations of process models to enhance their comprehension. Inf Syst 36:881–897.
139. Reijers H, Vanderfeesten I (2004) Cohesion and Coupling Metrics for Workflow Process Design. Bus Process Manag – Second Int Conf BPM 2004, Potsdam, Ger June 17–18, 2004 Proc 290–305.
140. Reinhartz-Berger I, Soffer P, Sturm A (2010) Extending the adaptability of reference models. IEEE Trans Syst Man, Cybern Part ASystems Humans 40:1045–1056.
141. Reinhartz-Berger I, Sturm A (2008) Enhancing UML models: a domain analysis approach. J Database Manag 19:74–94.
142. Richen A, Steinhorst A (2005) Standardization or harmonization? You need both. BP Trends Newsl.
143. Ripon SH, Talukder KH, Molla MKI (2003) Modelling variability for system families. Malaysian J Comput Sci 16:37–46.
144. Robson C (1997) Real world research. Oxford, Blackwell
145. Rogers JL (1990) Knowledge-Based Tool for Decomposing Complex Design Problems. J Comput Civ Eng 4:298–312.
146. Rosa L, Mendling J, Rosa M La (2012) Thresholds for Error Probability Measures of Business Process Models. J Syst Softw 85:1188–1197.
147. Rosa M La, Aalst WMP Van Der, Dumas M, Hofstede AHM (2009) Variability Modeling for Questionnaire-based System Configuration. Transit 8:1–29.
148. Rosa M La, Dumas M, Mendling J, Gottschalk F (2008) Beyond Control-Flow: Extending Business Process Con guration to Resources and Objects. Business. Springer, pp 199–215
149. Rosa M La, Dumas M, La Rosa M, Ter Hofstede AHM, Mendling J (2011) Configurable multi-perspective business process models. Inf Syst 36:313–340.
150. La Rosa M, Dumas M, Uba R, Dijkman RM (2013) Business Process Model Merging: An Approach to Business Process Consolidation. ACM Trans Softw Eng Methodol 22:11–11:42.

151. La Rosa M, Dumas M, Uba R, Dijkman RM (2013) Business Process Model Merging: An Approach to Business Process Consolidation. ACM Trans Softw Eng Methodol 22:1–42.
152. La Rosa M, Lux J, Seidel S, Dumas M, ter Hofstede AHM (2006) Questionnaire-driven Configuration of Reference Process Models. Adv Inf Syst Eng 19th Int Conf CAiSE 2007, Trondheim, Norway, June 11–15, 2007 Proc 424–438.
153. Rosemann M, Aalst W van der (2007) A configurable reference modelling language. Inf Syst 32:1–23.
154. Rosenkranz C, Seidel S, Mendling J, Schaefermeyer M, Recker J (2010) Towards a framework for business process standardization. Lect. Notes Bus. Inf. Process. Vol. 43. Springer, pp 53–63
155. Rummler G a, Brache AP (1995) Improving Performance: How to Manage the White Space on the Organization Chart. Jossey-Bass
156. Runeson P, Höst M (2009) Guidelines for conducting and reporting case study research in software engineering. Empir Softw Eng 14:131–164.
157. Runeson P, Host M, Rainer A, Regnell B (2012) Case Study Research in Software Engineering: Guidelines and Examples. Wiley
158. Sadiq S, Governatori G (2010) Managing regulatory compliance in business processes. In: Brocke J vom, Rosemann M (eds) Handb. Bus. Process Manag. 2. Springer Berlin Heidelberg, pp 159–175
159. Saidani O, Nurcan S (2007) Towards Context Aware Business Process Modelling. 8th Work Bus Process Model Dev Support 9.
160. Schäfermeyer M, Grgecic D, Rosenkranz C (2010) Factors Influencing Business Process Standardization : A Multiple Case Study. 43rd Hawaii Int. Conf. Syst. Sci. IEEE, pp 1–10
161. Schäfermeyer M, Rosenkranz C (2011) To Standardize or not to Standardize? – Understanding the Effect of Business Process Complexity on Business Process Standardization. ECIS 2011 Proc.
162. Schäfermeyer M, Rosenkranz C, Holten R (2012) The Impact of Business Process Complexity on Business Process Standardization. Bus Inf Syst Eng 4:261–270. doi: 10.1007/s12599-012-0224-6
163. Scheer AW, Thomas O, Adam O (2005) Process Modeling using Event-Driven Process Chains. Process. Inf. Syst. Bridg. People Softw. through Process Technol. John Wiley & Sons Inc., Hoboken, New Jersey, pp 119–145
164. Schnieders A, Puhlmann F (2006) Variability Mechanisms in E-Business Process Families. BIS. pp 583–601
165. Schnieders A, Puhlmann F (2007) Variability modeling and product derivation in e-business process families. Technol Bus Inf Syst 85:63–74.
166. Schunselaar DMM, Verbeek E, Van Der Aalst WMP, Raijers H a. (2012) Creating sound and reversible configurable process models using CoSeNets. Lect. Notes Bus. Inf. Process. Springer, pp 24–35
167. Sharp A, McDermott P (2009) Workflow Modeling: Tools for Process Improvement and Applications Development. Artech House

168. Simidchieva BI, Clarke L a., Osterweil LJ (2007) Representing process variation with a process family. Int. Conf. Softw. Process. ICSP 2007, Minneapolis, MN, USA, May 19–20, 2007. Proc. Springer, pp 109–120
169. Smirnov S, Dijkman R, Mendling J, Weske M (2010) Meronymy-based aggregation of activities in business process models. Lect Notes Comput Sci (including Subser Lect Notes Artif Intell Lect Notes Bioinformatics) 6412 LNCS:1–14.
170. Smirnov S, Reijers H a., Weske M, Nugteren T (2012) Business process model abstraction: A definition, catalog, and survey. Distrib Parallel Databases 30:63–99.
171. Smith EE, Medin DL (1981) Categories and concepts. Harvard University Press
172. Smith RP, Morrow J (1999) Product development process modeling. Des Stud 20:237–261.
173. Teichert T, Shehu E, von Wartburg I (2008) Customer segmentation revisited: The case of the airline industry. Transp Res Part A Policy Pract 42:227–242.
174. Torres V, Zugal S, Weber B, Reichert M, Ayora C, Pelechano V (2013) A qualitative comparison of approaches supporting business process variability. Lect Notes Bus Inf Process 132 LNBIP:560–572.
175. Torres V, Zugal S, Weber B, Reichert M, Ayora C, Pelechano V (2013) A qualitative comparison of approaches supporting business process variability. Lect. Notes Bus. Inf. Process. Springer, pp 560–572
176. Tregear R (2010) Business Process Standardization. Handb. Bus. Process Manag. 2. pp 307–327
177. Tsiptsis K, Chorianopoulos A (2010) Data Mining Techniques in CRM.
178. Turetken O, Demirors O (2011) Plural: A decentralized business process modeling method. Inf Manag 48:235–247.
179. Uba R, Dumas M (2011) Clone detection in repositories of business process models. Bus. Process Manag. LNCS. Springer Berlin Heidelberg, pp 302–318
180. Uba R, Dumas M, García-Bañuelos L, La Rosa M (2011) Clone detection in repositories of business process models. Bus Process Manag LNCS 6896:248–264.
181. Ungan, C. M (2006) Standardization through process documentation. Bus Process Manag J 12:135–148.
182. Vanderfeesten I, Cardoso J, Reijers H a. (2007) A weighted coupling metric for business process models. CEUR Workshop Proc 247:41–44.
183. Vanderfeesten I, Reijers H a., van der Aalst WMP (2008) Evaluating workflow process designs using cohesion and coupling metrics. Comput Ind 59:420–437.
184. Vanderfeesten I, Reijers H a., Mendling J, Van Der Aalst WMP, Cardoso J (2008) On a quest for good process models: The cross-connectivity metric. Lect Notes Comput Sci (including Subser Lect Notes Artif Intell Lect Notes Bioinformatics) 5074 LNCS:480–494.

185. Vanhatalo J, Völzer H, Koehler J (2008) The Refined Process Structure Tree Motivation : BPMN to BPEL Translation. Management
186. Wagemakers T (2009) Configurable Process Models : A municipality case study. Adv. Inf. Syst.
187. Weber B, Reichert M, Mendling J, Reijers HA (2011) Refactoring large process model repositories.pdf. Comput Ind 62:5:467–486.
188. Wedel M, Kamakura WA (2000) Market segmentation: Conceptual and methodological foundations.
189. Weiss DM (1999) Software product-line engineering: a family-based software development process.
190. Weske M (2010) Business process management: concepts, languages, architectures. Springer
191. Westerberg AW, Subrahmainan E, Reich Y, Konda S (1997) Designing the process design process. Comput Chem Eng 21:S1–S9.
192. White S a (2004) Introduction to BPMN. BPTrends 1–11.
193. Wohed P, Van Der Aalst WMP, Dumas M, Ter Hofstede AHM, Russell N (2005) Pattern-based analysis of the control-flow perspective of UML activity diagrams. Lect Notes Comput Sci (including Subser Lect Notes Artif Intell Lect Notes Bioinformatics) 3716 LNCS:63–78.
194. Wolter C, Schaad A (2007) Modeling of Task-Based Authorization Constraints in BPMN. In: Alonso G, Dadam P, Rosemann M (eds) Bus. Process Manag. Springer Berlin Heidelberg, pp 64–79
195. Yin RK (2009) Case study research: Design and methods. SAGE Publications
196. Young FW, Hamer RM (1987) Multidimensional scaling: History, theory, and applications. Erlbaum, New York
197. Zugal S, Pinggera J, Weber B, Mendling J, Reijers H a. (2011) Assessing the impact of hierarchy on model Understandability – A cognitive perspective. CEUR Workshop Proc. Springer, pp 18–27

KOKKUVÕTE (SUMMARY IN ESTONIAN)

Alamprotsessidest, protsesside variatsioonidest ja nendevahelisest koosmõjust: Integreeritud "jaga ja valitse" meetod äriprotsesside ja nende variatsioonide modelleerimiseks

Igat organisatsiooni võib vaadelda kui süsteemi, mis rakendab äriprotsesse väärtuste loomiseks. Suurtes organisatsioonides on tavapärane esitada äriprotsesse kasutades protsessimudeleid, mida kasutatakse erinevatel eesmärkidel nagu näiteks sisekommunikatsiooniks, koolitusteks, protsesside parendamiseks ja infosüsteemide arendamiseks. Arvestades protsessimudelite multifunktsionaalset olemust tuleb protsessimudeleid koostada selliselt, et see võimaldab nendest arusaamist ning haldamist erinevate osapoolte poolt.

Modelleerimise seisukohast võib äriprotsesse jagada kahel viisil. Ühest küljest võib protsessi jagada alamprotsessideks selliselt, et nende kokkuliitmine annab tulemuseks terve protsessi. Teisest küljest võib protsessi jagada variatsioonideks selliselt, et iga variatsioon esitab alamosa võimalikest protsessi käivitamistest. Sellist lähenemist kasutades moodustub terve protsessi kirjeldus tema variatsioonide ühendist. Mõlemal lähenemisel on omad eelised ning puudused. Näiteks on suuri protsessimudeleid, mis kätkevad endas kõikvõimalikke variatsioone, raske hoomata ning neid töös kasutada. Samas modelleerides igat variatsiooni eraldi protsessimudelina on tulemuseks suur hulk eraldiseisvaid protsessimudeleid, mida on keerukas hallata ning milles on palju liiasust kuna ühe protsessi samad fragmendid on esindatud erinevates mudelites.

Nimetatud eesmärgi saavutamiseks on laialdaselt aksepteeritud lähenemine, et keerukat äriprotsessi ei peaks koondama ühte suurde mudelisse, vaid pigem hulka väiksematesse mudelitesse jälgides "jaga ja valitse" põhimõtet. See soovitus kehtib eriti äriprotsessidele, millel on erinevaid variatsioone, nt order-to-cash protsess, mis varieerub sõltuvalt geograafilisest regioonist, kliendi liigist või toote tüübist.

Käesoleva töö positsioneerimiseks sai läbi viidud kirjanduse uuring variatsioonide haldamise kohta äriprotsessides ja protsessimudelites (peatükk 2). Erinevate lähenemiste analüüs tõi välja, et lähenemiste puhul võiks eristada laias laastus kahte dimensiooni. Esiteks pakkusid vaadeldud lähenemised välja mehhanismid kas protsessimudelite fragmenteerimiseks või konsolideerimiseks. Sellest johtub, et variatsioone hallatakse läbi operatsioonide, mis kas vähendavad (kitsendavad või standardiseerivad) või suurendavad (laiendavad või kohendavad) mudelite arvu. Teiseks dimensiooniks on muudatuste subjekt – on see äriprotsess või protsessimudel. Mõned lähenemised on disanitud äriprotsesside loomiseks või muutmiseks samas kui teised on olemasolevate protsessimudelite parendamiseks.

Uuring tõi välja, et olgugi, et äriprotsesside mudelite variatsioonide haldamiseks on palju lähenemisi, neil kõigil on omad puudused. Esiteks nad

eeldavad, et protsessimudelid on samas modelleerimiskeeles ja/või kasutavad sama notatsiooni ja teiseks ei hooli nad variatsioonide ajenditest protsessimudelite struktureerimisel. Eeldus sama modelleerimiskeele ning notatsiooni kasutamise kohta ei ole paraku realistlik paljudes praktilistes stsenaariumites, kus erinevate variantide mudelid ei pruugi olla kas käepärast või on neid modelleeritud eri tiimide poolt kasutades eri keeli ja/või notatsioone. Lisaks pole need mudelid arvatavasti modelleeritud läbivalt samal granulaarsuse tasemel.

Käesolev doktoritöö käsitleb seda nishi pakkudes välja integreeritud dekompositsioonist ajendatud meetodi äriprotsesside modelleerimiseks koos nende variatsioonidega (peatükk 5). Meetodi kandvaks ideeks on järkjärguline äriprotsessi ja selle variatsioonide dekomponeerimine alamprotsessideks. Igal dekompositsiooni tasemel ning iga alamprotsessi jaoks määratletakse esmalt kas vastavat alamprotsessi tuleks modelleerida konsolideeritud moel (üks alamprotsessi mudel kõikide või osade variatsioonide jaoks) või fragmenteeritud moel (üks alamprotsess ühe variatsiooni jaoks). Sel moel kasutades ülalt-alla lähenemist viilutatakse ja tükeldatakse äriprotsess väiksemateks osadeks.

Protsessimudeleid dekomponeerides võib rakendada erinevaid heuristikaid (peatükk 3). Samas, praktikas rahuldavad vaid kaks heuristikat kriteeriumid tarvilikkuse ja piisavuse osas protsessimudelite dekomponeerimisel. Need heuristikad baseeruvad loogilistel äriprotsesside murdepunktidel ning äriprotsesside tegevuste vahel jagatud andmeobjektidel. Nende heuristikate empiiriline hindamine on näidanud, et konkreetse heuristika valik ei mõjuta protsessimudelite arusaadavust.

Igal äriprotsessi variatsioonil on oma juurpõhjus, mis pärineb ärist endast ja põhjustab protsesside käivitamisel erisusi (peatükk 4). Need juurpõhjused jagatakse viide kategooriasse – ajendid kliendist, tootest, operatiivsetest põhjustest, turust ja ajast. Teine parameeter on erinevuste hulk viisides (tegevuste järjekord, tulemuste väärtused jms) kuidas variatsioonid oma väljundit toodavad.

Variatsioonide modelleerimine sõltub lisaks veel äriprotsesside erinevustest, nagu näiteks tegevused, objektid ja ressursid. Kui äriprotsessid erinevad üksteisest oluliselt, on mugavam neid eraldi hoida. Samas, kui protsessid on väga sarnased, mudeldatakse neid koos, et vähendada duplitseerimist.

Käesolevas töös esitatud meetod on valideeritud kahes praktilises juhtumiuuringus (peatükk 6). Kui esimeses juhtumiuuringus on põhirõhk olemasolevate protsessimudelite konsolideerimisel, siis teises protsessimudelite avastamisel. Sel moel rakendatakse meetodit kahes eri kontekstis kahele üksteisest eristatud juhtumile. Mõlemas juhtumiuuringus tootis meetod protsessimudelite hulgad, milles oli liiasust kuni 50% vähem võrreldes tavapäraste meetoditega jättes samas mudelite keerukuse nendega võrreldes enamvähem samale tasemele.

CURRICULUM VITAE

Name: Fredrik Milani
Date of Birth: 16.11.1974
Citizenship: Swedish

Education

2011–2015	University of Tartu, Faculty of Mathematics and Computer Science Doctoral Studies, Specialty: Computer Science
1999–2000	University of Stockholm, Sweden Master Degree in Business Administration and Management
1995–1999	University of Stockholm, Sweden Bachelors Degree in Business Administration and Management
1990–1993	Jenny Nyströmsskolan, Kalmar, Sweden, Secondary Education Natural Sciences Programme
1981–1990	Barkestorpsskolan, Kalmar, Rootsi, Primary Education

Work Experience

2006–2011	Handelsbanken Capital Markets, Stockholm, Sweden Manager of Derivatives Operations Business Support
2004–2006	Handelsbanken Capital Markets, Stockholm, Sweden Senior Business Analyst
2000–2004	Handelsbanken Capital Markets, Stockholm, Sweden Business Analyst

ELULOOKIRJELDUS

Nimi: Fredrik Milani
Sünniaeg: 16.11.1974
Kodakondsus: Rootsi

Haridus

2011–2015 Tartu Ülikool,
Matemaatika-informaatikateaduskond,
Doktoriõpe, eriala: arvutiteadus

1999–2000 Stockholmi Ülikool, Rootsi
Magistriõpe, eriala: ärijuhtimine

1995–1999 Bakalaureusekraad, eriala: ärijuhtimine

1990–1993 Jenny Nyströmsskolan, Kalmar, Rootsi, keskharidus,
Loodusteaduste õppekava

1981–1990 Barkestorpsskolan, Kalmar, Rootsi, algharidus

Teenistuskäik

2006–2011 Handelsbanken Capital Markets, Stockholm, Sweden
Juhataja, derivaatide toimingute äritugi

2004–2006 Handelsbanken Capital Markets, Stockholm, Sweden
Vanemärianalüütik

2000–2004 Handelsbanken Capital Markets, Stockholm, Sweden
Ärianalüütik

DISSERTATIONES MATHEMATICAE UNIVERSITATIS TARTUENSIS

1. **Mati Heinloo.** The design of nonhomogeneous spherical vessels, cylindrical tubes and circular discs. Tartu, 1991, 23 p.
2. **Boris Komrakov.** Primitive actions and the Sophus Lie problem. Tartu, 1991, 14 p.
3. **Jaak Heinloo.** Phenomenological (continuum) theory of turbulence. Tartu, 1992, 47 p.
4. **Ants Tauts.** Infinite formulae in intuitionistic logic of higher order. Tartu, 1992, 15 p.
5. **Tarmo Soomere.** Kinetic theory of Rossby waves. Tartu, 1992, 32 p.
6. **Jüri Majak.** Optimization of plastic axisymmetric plates and shells in the case of Von Mises yield condition. Tartu, 1992, 32 p.
7. **Ants Aasma.** Matrix transformations of summability and absolute summability fields of matrix methods. Tartu, 1993, 32 p.
8. **Helle Hein.** Optimization of plastic axisymmetric plates and shells with piece-wise constant thickness. Tartu, 1993, 28 p.
9. **Toomas Kiho.** Study of optimality of iterated Lavrentiev method and its generalizations. Tartu, 1994, 23 p.
10. **Arne Kokk.** Joint spectral theory and extension of non-trivial multiplicative linear functionals. Tartu, 1995, 165 p.
11. **Toomas Lepikult.** Automated calculation of dynamically loaded rigid-plastic structures. Tartu, 1995, 93 p, (in Russian).
12. **Sander Hannus.** Parametrical optimization of the plastic cylindrical shells by taking into account geometrical and physical nonlinearities. Tartu, 1995, 74 p, (in Russian).
13. **Sergei Tupailo.** Hilbert's epsilon-symbol in predicative subsystems of analysis. Tartu, 1996, 134 p.
14. **Enno Saks.** Analysis and optimization of elastic-plastic shafts in torsion. Tartu, 1996, 96 p.
15. **Valdis Laan.** Pullbacks and flatness properties of acts. Tartu, 1999, 90 p.
16. **Märt Põldvere.** Subspaces of Banach spaces having Phelps' uniqueness property. Tartu, 1999, 74 p.
17. **Jelena Ausekle.** Compactness of operators in Lorentz and Orlicz sequence spaces. Tartu, 1999, 72 p.
18. **Krista Fischer.** Structural mean models for analyzing the effect of compliance in clinical trials. Tartu, 1999, 124 p.
19. **Helger Lipmaa.** Secure and efficient time-stamping systems. Tartu, 1999, 56 p.
20. **Jüri Lember.** Consistency of empirical k-centres. Tartu, 1999, 148 p.
21. **Ella Puman.** Optimization of plastic conical shells. Tartu, 2000, 102 p.
22. **Kaili Müürisep.** Eesti keele arvutigrammatika: süntaks. Tartu, 2000, 107 lk.

23. **Varmo Vene.** Categorical programming with inductive and coinductive types. Tartu, 2000, 116 p.
24. **Olga Sokratova.** Ω-rings, their flat and projective acts with some applications. Tartu, 2000, 120 p.
25. **Maria Zeltser.** Investigation of double sequence spaces by soft and hard analitical methods. Tartu, 2001, 154 p.
26. **Ernst Tungel.** Optimization of plastic spherical shells. Tartu, 2001, 90 p.
27. **Tiina Puolakainen.** Eesti keele arvutigrammatika: morfoloogiline ühestamine. Tartu, 2001, 138 p.
28. **Rainis Haller.** *M(r,s)*-inequalities. Tartu, 2002, 78 p.
29. **Jan Villemson.** Size-efficient interval time stamps. Tartu, 2002, 82 p.
30. **Eno Tõnisson.** Solving of expession manipulation exercises in computer algebra systems. Tartu, 2002, 92 p.
31. **Mart Abel.** Structure of Gelfand-Mazur algebras. Tartu, 2003. 94 p.
32. **Vladimir Kuchmei.** Affine completeness of some ockham algebras. Tartu, 2003. 100 p.
33. **Olga Dunajeva.** Asymptotic matrix methods in statistical inference problems. Tartu 2003. 78 p.
34. **Mare Tarang.** Stability of the spline collocation method for volterra integro-differential equations. Tartu 2004. 90 p.
35. **Tatjana Nahtman.** Permutation invariance and reparameterizations in linear models. Tartu 2004. 91 p.
36. **Märt Möls.** Linear mixed models with equivalent predictors. Tartu 2004. 70 p.
37. **Kristiina Hakk.** Approximation methods for weakly singular integral equations with discontinuous coefficients. Tartu 2004, 137 p.
38. **Meelis Käärik.** Fitting sets to probability distributions. Tartu 2005, 90 p.
39. **Inga Parts.** Piecewise polynomial collocation methods for solving weakly singular integro-differential equations. Tartu 2005, 140 p.
40. **Natalia Saealle.** Convergence and summability with speed of functional series. Tartu 2005, 91 p.
41. **Tanel Kaart.** The reliability of linear mixed models in genetic studies. Tartu 2006, 124 p.
42. **Kadre Torn.** Shear and bending response of inelastic structures to dynamic load. Tartu 2006, 142 p.
43. **Kristel Mikkor.** Uniform factorisation for compact subsets of Banach spaces of operators. Tartu 2006, 72 p.
44. **Darja Saveljeva.** Quadratic and cubic spline collocation for Volterra integral equations. Tartu 2006, 117 p.
45. **Kristo Heero.** Path planning and learning strategies for mobile robots in dynamic partially unknown environments. Tartu 2006, 123 p.
46. **Annely Mürk.** Optimization of inelastic plates with cracks. Tartu 2006. 137 p.
47. **Annemai Raidjõe.** Sequence spaces defined by modulus functions and superposition operators. Tartu 2006, 97 p.

48. **Olga Panova.** Real Gelfand-Mazur algebras. Tartu 2006, 82 p.
49. **Härmel Nestra.** Iteratively defined transfinite trace semantics and program slicing with respect to them. Tartu 2006, 116 p.
50. **Margus Pihlak.** Approximation of multivariate distribution functions. Tartu 2007, 82 p.
51. **Ene Käärik.** Handling dropouts in repeated measurements using copulas. Tartu 2007, 99 p.
52. **Artur Sepp.** Affine models in mathematical finance: an analytical approach. Tartu 2007, 147 p.
53. **Marina Issakova.** Solving of linear equations, linear inequalities and systems of linear equations in interactive learning environment. Tartu 2007, 170 p.
54. **Kaja Sõstra.** Restriction estimator for domains. Tartu 2007, 104 p.
55. **Kaarel Kaljurand.** Attempto controlled English as a Semantic Web language. Tartu 2007, 162 p.
56. **Mart Anton.** Mechanical modeling of IPMC actuators at large deformations. Tartu 2008, 123 p.
57. **Evely Leetma.** Solution of smoothing problems with obstacles. Tartu 2009, 81 p.
58. **Ants Kaasik.** Estimating ruin probabilities in the Cramér-Lundberg model with heavy-tailed claims. Tartu 2009, 139 p.
59. **Reimo Palm.** Numerical Comparison of Regularization Algorithms for Solving Ill-Posed Problems. Tartu 2010, 105 p.
60. **Indrek Zolk.** The commuting bounded approximation property of Banach spaces. Tartu 2010, 107 p.
61. **Jüri Reimand.** Functional analysis of gene lists, networks and regulatory systems. Tartu 2010, 153 p.
62. **Ahti Peder.** Superpositional Graphs and Finding the Description of Structure by Counting Method. Tartu 2010, 87 p.
63. **Marek Kolk.** Piecewise Polynomial Collocation for Volterra Integral Equations with Singularities. Tartu 2010, 134 p.
64. **Vesal Vojdani.** Static Data Race Analysis of Heap-Manipulating C Programs. Tartu 2010, 137 p.
65. **Larissa Roots.** Free vibrations of stepped cylindrical shells containing cracks. Tartu 2010, 94 p.
66. **Mark Fišel.** Optimizing Statistical Machine Translation via Input Modification. Tartu 2011, 104 p.
67. **Margus Niitsoo**. Black-box Oracle Separation Techniques with Applications in Time-stamping. Tartu 2011, 174 p.
68. **Olga Liivapuu.** Graded q-differential algebras and algebraic models in noncommutative geometry. Tartu 2011, 112 p.
69. **Aleksei Lissitsin.** Convex approximation properties of Banach spaces. Tartu 2011, 107 p.
70. **Lauri Tart.** Morita equivalence of partially ordered semigroups. Tartu 2011, 101 p.

71. **Siim Karus.** Maintainability of XML Transformations. Tartu 2011, 142 p.
72. **Margus Treumuth.** A Framework for Asynchronous Dialogue Systems: Concepts, Issues and Design Aspects. Tartu 2011, 95 p.
73. **Dmitri Lepp.** Solving simplification problems in the domain of exponents, monomials and polynomials in interactive learning environment T-algebra. Tartu 2011, 202 p.
74. **Meelis Kull.** Statistical enrichment analysis in algorithms for studying gene regulation. Tartu 2011, 151 p.
75. **Nadežda Bazunova.** Differential calculus $d^3 = 0$ on binary and ternary associative algebras. Tartu 2011, 99 p.
76. **Natalja Lepik.** Estimation of domains under restrictions built upon generalized regression and synthetic estimators. Tartu 2011, 133 p.
77. **Bingsheng Zhang.** Efficient cryptographic protocols for secure and private remote databases. Tartu 2011, 206 p.
78. **Reina Uba.** Merging business process models. Tartu 2011, 166 p.
79. **Uuno Puus.** Structural performance as a success factor in software development projects – Estonian experience. Tartu 2012, 106 p.
80. **Marje Johanson.** $M(r, s)$-ideals of compact operators. Tartu 2012, 103 p.
81. **Georg Singer.** Web search engines and complex information needs. Tartu 2012, 218 p.
82. **Vitali Retšnoi.** Vector fields and Lie group representations. Tartu 2012, 108 p.
83. **Dan Bogdanov.** Sharemind: programmable secure computations with practical applications. Tartu 2013, 191 p.
84. **Jevgeni Kabanov.** Towards a more productive Java EE ecosystem. Tartu 2013, 151 p.
85. **Erge Ideon**. Rational spline collocation for boundary value problems. Tartu, 2013, 111 p.
86. **Esta Kägo**. Natural vibrations of elastic stepped plates with cracks. Tartu, 2013, 114 p.
87. **Margus Freudenthal.** Simpl: A toolkit for Domain-Specific Language development in enterprise information systems. Tartu, 2013, 151 p.
88. **Boriss Vlassov.** Optimization of stepped plates in the case of smooth yield surfaces. Tartu, 2013, 104 p.
89. **Elina Safiulina.** Parallel and semiparallel space-like submanifolds of low dimension in pseudo-Euclidean space. Tartu, 2013, 85 p.
90. **Raivo Kolde.** Methods for re-using public gene expression data. Tartu, 2014, 121 p.
91. **Vladimir Šor.** Statistical Approach for Memory Leak Detection in Java Applications. Tartu, 2014, 155 p.
92. **Naved Ahmed.** Deriving Security Requirements from Business Process Models. Tartu, 2014, 171 p.
93. **Kerli Orav-Puurand.** Central Part Interpolation Schemes for Weakly Singular Integral Equations. Tartu, 2014, 109 p.

94. **Liina Kamm.** Privacy-preserving statistical analysis using secure multi-party computation. Tartu, 2015, 201 p.
95. **Kaido Lätt.** Singular fractional differential equations and cordial Volterra integral operators. Tartu, 2015, 93 p.
96. **Oleg Košik.** Categorical equivalence in algebra. Tartu, 2015, 84 p.
97. **Kati Ain.** Compactness and null sequences defined by ℓ_p spaces. Tartu, 2015, 90 p.
98. **Helle Hallik.** Rational spline histopolation. Tartu, 2015, 100 p.
99. **Johann Langemets.** Geometrical structure in diameter 2 Banach spaces. Tartu, 2015, 130 p.
100. **Abel Armas Cervantes.** Diagnosing Behavioral Differences between Business Process Models. Tartu, 2015, 193 p.