Composing workflow activities

automated support on the basis of data-flow structures

van der Aa, J.H.

Award date:
2013

Technische Universiteit
Eindhoven
University of Technology

Department of Mathematics and Computer Science
Architecture of Information Systems Research Group

MASTER'S THESIS

Composing Workflow Activities

Automated Support on the Basis of Data-flow Structures

by
J.H. van der Aa

Supervisors:
prof.dr.ir. H.A. Reijers
dr.ir. I.T.P. Vanderfeesten
dr.ir. D. Fahland

Eindhoven, April 2013

ABSTRACT

The proper composition of activities is important for the design of efficient and understandable workflow processes. Due to a lack of available support, activity composition is currently a time-consuming, manual task that requires expertise and case knowledge. This thesis presents an automated approach that addresses this issue. The approach utilizes the data-flow underlying a workflow process to determine the importance and semantic relatedness of the various, elementary data-processing steps. Based on these aspects, fundamental guidelines are proposed to drive and objectify the task of activity composition in the context of workflow design. A quantitative evaluation indicates that the use of these guidelines leads to activities that closely resemble those designed by experienced modellers. The guidelines have been implemented in software that has been made freely available. This enables any modeller to generate proper activities for a given workflow, irrespective of their familiarity with the business process or the concepts of activity design.

ACKNOWLEDGMENTS

This master's thesis is the result of research that has been conducted at the Eindhoven University of Technology over the past seven months. I take this opportunity to thank those that helped and supported me during this process.

I am immensely grateful to my supervisors, Hajo Reijers and Irene Vanderfeesten, for their invaluable input and support throughout this entire project. Your collective efforts drove this research to a higher level. Furthermore, I want to thank Dirk Fahland for joining the assessment committee and for his detailed feedback.

My gratitude moreover reaches out to those that selflessly assisted me throughout the research process. I want to thank Gena Drahun (Perceptive Software) and Peter Ruijten for their input on interaction design, as well as Sergey Smirnov (SAP) for sharing his expertise on business process model abstraction. I also want to thank Eric Verbeek for his assistance during the development of the ProM-plugin. Furthermore, I am very thankful to Wim van de Crommert and Hector Diaz Garcia for their efforts to expand the quantitative evaluation.

I thank all my friends for the great times we had in Eindhoven. Special thanks go out to Wim and Jordy for making the office life more enjoyable. Also, to my former colleagues at Unipartners for their help and friendship.

My sincere appreciation goes out to my family for their continuous encouragement throughout my time as a student. Lastly, I want to thank Ilse, for all her support.

Han van der Aa
Eindhoven, April 2013

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

In the early 1990s, the fundamental rethinking and radical redesign of business processes emerged. Business Process Redesign (BPR) shifted processes away from Adam Smith's division of labour, in which simple tasks resulted in complex processes, towards re-engineered processes that are kept simple in order to meet contemporary demands of quality, flexibility and low cost [8]. The application of information technology (IT), and in particular of Process Aware Information Systems (PAIS), plays an important role in realizing these process improvements [35]. The continuous advancements in IT rapidly increase the ease with which information can be digitally stored and accessed. The impact of these new technologies has resulted in radical changes in information-intensive business processes. These processes, often found in the service industry, pursue the production of informational products. Typical examples of such informational products are a mortgage contract, requests for governmental support, or a commercial offer. Such processes shall be referred to as *workflows*. The structure of a workflow is for a major part dictated by the data-flow in the process. Consequently, the design of workflow processes is considerably more flexible than, for example, the design of manufacturing processes, which process physical parts [26]. A data-flow can be described by the numerous, elementary data processing steps that are involved in the computation of a desired end product. When designing a workflow, these elementary data processing steps are grouped into *activities* (or *tasks*). An activity is a *logical piece of work* within a process [32], which may thus comprise a number of elementary data processing steps. For example, the activity of calculating a mortgage amount may consist of entering the current interest rate, choosing the discount rate negotiated by the customer, and calculating the amortized amount of debt.

The focus of attention in this thesis is on the grouping of elementary data processing steps into activities. This act shall be referred to as *activity composition*. When composing activities for a workflow, one should carefully consider their size and contents, because properly designed activities can result in three kinds of advantages. Firstly, activities that have a proper size, i.e. are of the right *granularity*, provide a balance between an increased number of work hand-overs that results from many small activities, against the reduced flexibility caused by too many large activities [29, 36]. Secondly, activity composition can be used to increase the meaningfulness of activities for employees executing these [7]. Finally, techniques for activity compo-

sition are relevant for the topic of Business Process Model Abstraction (BPMA) [22]. In particular, given a detailed understanding of the data-flow in a process, activity composition techniques can be used to provide a quick overview of the process. This has been found to be a highly demanded use case for BPMA [30]. By contrast, improper activity composition can result in the averse, and hence, negative effects. Such improper composition may, for example, result in activities that are too large and therefore unworkable for employees [24, 36]. Incorrect activities can also lead to processes that contain deadlocks and therefore cannot be executed. It is thus important that activities are composed such that the undesirable effects are avoided, while the potential benefits of well-designed activities are attained. It is the goal of the research presented in this thesis to provide support for such proper activity composition.

This first chapter elaborates this research goal in Section 1.1. Afterwards, related work is considered in Section 1.2, and Section 1.3, finally, provides an overview of the remaining chapters.

1.1 RESEARCH GOAL

Due to the value of proper activity composition, there exists a desire for support that drives the design of activities [26, 36]. Because such support is currently not available [35], activity composition is a time-consuming task that requires expertise and case knowledge. For example, one must be familiar with concepts of task design in order to ensure that activities are of the proper size, represent meaningful tasks, and do not negatively affect execution efficiency. To design proper activities, a modeller must furthermore be well-acquainted with the elementary data processing steps in a workflow, and understand their inter-relations. Due to these requirements, activity composition is here considered to be a task that must be performed by *experts*. Still, even for experts, manual activity composition can be a time-consuming task due to the potential size and complexity of data-flows.

Activity composition is thus an important part of workflow design, but also one that lacks appropriate support. This issue is captured in the problem statement of this thesis.

PROBLEM STATEMENT. Activity composition is important for the design of efficient and understandable workflow processes, but due to a lack of support, the task is time-consuming and depends on expertise, as well as case knowledge.

This thesis presents two contributions to assess this problem. They are captured in two research goals. The first contribution is an approach that automatically generates activities for a given workflow. This is achieved through the design of guidelines that drive and ob-

jectify the task of activity composition. Although activity composition is a semantic operation, it is proposed that these guidelines approximate the design choices made by experienced modellers. Thereby, the automated approach represents an objective alternative to a subjective task. Through this contribution, activity composition no longer requires in-depth knowledge of the information processing steps in a workflow, nor familiarity with concepts of job design. Hence, the time-consuming, manual task of activity composition is replaced by an automated approach that can be applied by any process modeller. This contribution is defined in the first research goal.

RESEARCH GOAL 1. Design an approach that supports the task of proper activity composition so that it can be performed in a time-efficient manner, irrespective of case knowledge and expertise.

It is proposed that the automated approach generates activities which approximate the subjective design choices made my experienced modellers. This proposition, however, does not imply that the guidelines perfectly capture any modeller's preferences. For this reason, the second contribution of this thesis is the support for manual refinement of the generated activities. Through this support, the preferences of modellers can be incorporated in activities, while preserving the time-efficiency that is provided by a largely automated approach. This is captured in the second research goal.

RESEARCH GOAL 2. Extend the approach of Research goal 1 such that it supports the iterative refinement of activities based on a modeller's preferences.

1.2 RELATED WORK

The research presented in this thesis extends the state of the art in several ways. The interaction between the steps in a process and the data being processed is at the basis of a wide range of research efforts [28, 33, 34]. Yet, their exclusive focus is on the detection of data-flow errors. Although the importance of correct workflow processes is eminent, they do not asses how the interaction between the informational and the functional perspective of workflow processes can be used for the design of meaningful tasks. Job design literature does address the this meaningfulness. The Job Characteristics Model [7], for example, identifies *skill variety*, *task identity*, and *task significance* as the three characteristics that determine the psychological meaningfulness of a job. Still, the guidance provided through these models is rather abstract and does not rest on a detailed understanding of the data-flow perspective in a specific process. *Cohesion & Coupling* metrics [26, 36] do integrate insights from job design with the data-flow perspective of workflow processes. These metrics propose that the internal data-flow in activities should be highly cohesive, while, on

the contrary, the flow between individual activities should be limited. Cohesion & Coupling, however, are evaluation metrics and hence can only be retrospectively applied on previously composed activities. Finally, it is noted that the BPMA approach in [31] identifies clusters of worfklow tasks that are related through the data-flow. However, since this approach abstracts from the job design perspective, the clusters do not represent meaningful activities.

The state of the art is hence extended by proposing guidelines that ensure correct data processing, while combining job design insights with the data-flow perspective, to compose meaningful activities.

1.3 OUTLINE

The research in this paper consists of six subsequent phases; these are depicted in the research design presented in Figure 1.1. The remainder of this thesis is structured such that each of the upcoming chapters describes one of the research phases.

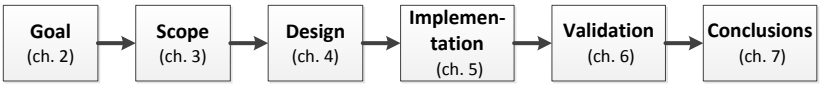


Figure 1.1: Research Design

1.3.1 *Goal*

The research goal is made operational in Chapter 2 by considering activity composition in the context of Product Based Workflow Design. The chapter furthermore motivates the research goal by performing activity composition on a running example. This example illustrates the differences between proper and improper activity designs.

1.3.2 *Scope*

After the goal of activity composition has been motivated, Chapter 3 defines the scope of this research. This chapter describes a broad variety of *factors* that can be considered during activity composition. The goal of this chapter is to identify those factors that are suitable to be incorporated in automated activity composition guidelines.

1.3.3 *Design*

Chapter 4 describes several activity composition guidelines that are based on the factors identified in Chapter 3. These guidelines are

designed such that they can be applied in an objective manner. The most important guidelines are designed such that they require only a minimal amount of process information. Additional guidelines are proposed that are able to further refine activities when more process information is available.

1.3.4 *Implementation*

Chapter 5 describes an algorithm that implements the proposed composition guidelines. It fully automatically generates a set of activities for a given PDM. The chapter furthermore introduces freely available software that implements the proposed algorithm. This software also supports the manual refinement of activities based on one's personal preferences. It thereby realizes both research goals of this thesis.

1.3.5 *Validation*

Chapter 6 presents a quantitative evaluation of activity designs that adhere to the proposed guidelines. This validation compares automatically generated activity designs with a set of activity designs derived from literature. These activity designs are all manually composed by modellers that are experienced in PBWD and are well-acquainted with the cases for which they designed the solutions. The validation supports the proposition that the objective guidelines are good approximations of design choices made by experienced modellers.

1.3.6 *Conclusions*

Finally, Chapter 7 concludes the paper with a discussion of the results of this research. The chapter furthermore considers limitations of the presented research, and completes the thesis with directions for future research.

2 ACTIVITY COMPOSITION

Activity composition is the act of grouping together atomic information processing steps in order to compose activities that represent *logical pieces of work*. The proper composition of activities results in increased process efficiency, tasks that are meaningful for the workflow users that execute them, and increases understandability of process models. However, inconsiderate activity composition can result in averse, and hence negative, effects. The task of activity composition is here made operational by considering it in the context of Product Based Workflow Design (PBWD), a methodology for the radical redesign of workflows [27]. For this purpose, structural data-flow relations are captured in a Product Data Model (PDM). Nonetheless, the guidelines presented in this thesis can be transferred to comparable data-flow specifications, such as the data flow matrices of [33]. The PBWD methodology is especially suited for this research due to its objectivity, and because the structure of a PDM ensures data-flow correctness [24].

This chapter sets out to further motivate the goal of activity composition. Section 2.1 first describes the PBWD methodology. Section 2.2 introduces a running example that is referenced throughout the remainder of this thesis. Finally, Section 2.3 presents and compares two alternative process designs, resultant from proper and improper activity composition.

2.1 PRODUCT BASED WORKFLOW DESIGN

PBWD is a methodology for the radical redesign of workflow processes. In PBWD, the central concept in the design process is the workflow product, i.e. the informational product to be created in the process [35, p.8]. The structure of this workflow product is captured in a PDM. A PDM describes the flow of information through a number of elementary processing steps. These steps, called *operations*, process information, starting from the data that is provided as input to the process, until the desired informational product has been computed. Example 2.1 explains the PDM by presenting a small example.

EXAMPLE 2.1 (MAXIMUM MORTGAGE): Figure 2.1 depicts a PDM related to the calculation of the maximum mortgage amount a client is able. The example is derived from [35, p.43]. The PDM contains eight *data elements*, which are depicted as labelled circles in Figure 2.1. Each data element represents a certain piece of information. They are described in Table 2.1. Data element A is the *root* data element of the

PDM; it represents the desired goal of the process. A value for this root element is computed by executing elementary information processing steps, called *operations*. These operations are depicted as black dots in the PDM. Each operation produces a value for exactly one data element, i.e. its *output element*, illustrated by a directed arc. Each operation requires a set of zero or more *input elements*. An operation can only be executed if *all* its input elements have been previously computed. For example, data element C can only be computed if data elements F, G, and H are available to the operation. Operations that do not require input elements are referred to as *leaf operations*. The data elements produced by such operations represent data that is provided as input to the process. For example, data element G, the gross income per year, is provided by the applicant. Finally, the figure shows that element A is produced by three operations. These operations represent *alternative operations*. Generally, these operations are mutual exclusive alternatives to compute a data element.

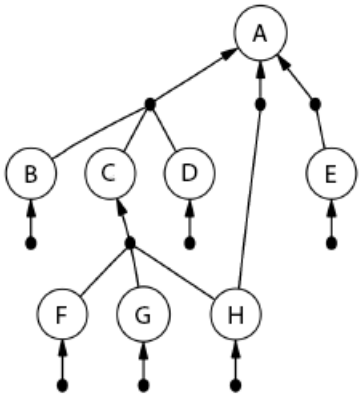


Figure 2.1: PDM of the mortgage example

ID	DESCRIPTION
A	Maximum mortgage
B	Percentage of interest
C	Annual budget to be spent on mortgage
D	Term of mortgage
E	Previous mortgage offer
F	Percentage of income to be spent on mortgage
G	Gross income per year
H	Credit registration

Table 2.1: Data elements in the mortgage example

2.2 RUNNING EXAMPLE

To motivate the goal of activity composition and add meaning to several upcoming concepts, a running example is presented that is referenced in the remainder of this paper. The running example considers the process that deals with requests for governmental student grants in the Netherlands. The example, introduced in [35, 36], is designed to introduce certain concepts in PBWD and is therefore specifically suitable as a running example. The presented process is a simplified version of the actual procedure as implemented by the Dienst Uitvo-

ering Onderwijs[1], the governmental institution responsible for the assessment of student grant requests.

Figure 2.2 presents the PDM of the *student grants* example. Its data elements are described in Table 2.2. A student grant comprises four types of grant: i) a basic grant (i40), ii) a supplementary grant (i39), iii) a loan (i41), and iv) credit for tuition fees (i43). The basic grant is assigned to each applicant who is eligible to receive a grant (i27). For this, an applicant must have the Dutch nationality (i21) and may not be older than thirty (i23). The remaining three types of grants must be specifically requested by applicants. Assignment of these grants depends on extended requirements. The root element (i42) represents the total grant assigned to an applicant. This is either the sum of the values for the four types of grant, or zero in case the applicant is deemed non-eligible. Finally, it is noted that all eleven leaf data elements represent information that is retrieved from a student's application. The interested reader is referred to [35, p.193] for a more detailed description of the student grants case.

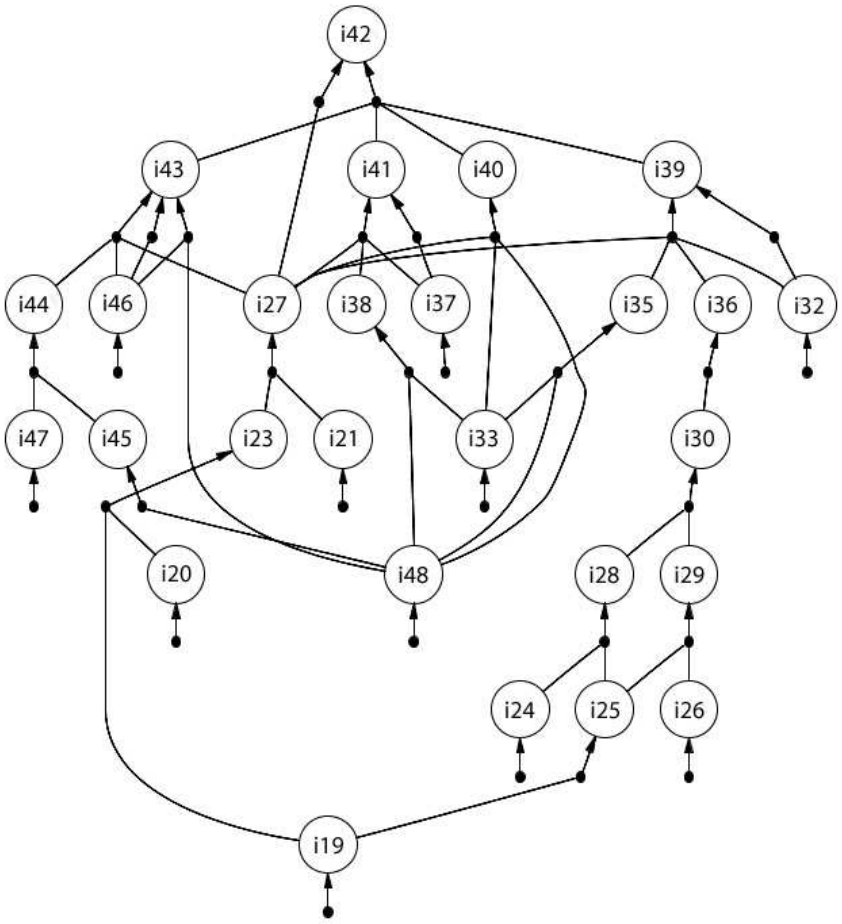


Figure 2.2: PDM of the student grants example

1 www.duo.nl

ID	DESCRIPTION
i19	Date of request
i20	Birth date of applicant
i21	Nationality of applicant
i23	Age of applicant
i24	Social Security Number of father of applicant
i25	Reference year for tax authority
i26	Social Security Number of mother of applicant
i27	Applicant has the right to receive a student grant
i28	Income of father of applicant
i29	Income of mother of applicant
i30	Income of parents of applicant
i31	Applicant has the right to receive a supplementary grant
i32	Applicant has requested a supplementary grant
i33	Living situation of applicant
i35	Maximum amount of supplementary grant
i36	Parental contribution
i37	Requested amount of loan by applicant
i38	Maximum amount of loan
i39	Amount of supplementary grant assigned to applicant
i40	Amount of basic grant assigned to applicant
i41	Amount of loan assigned to applicant
i42	Total amount of student grant assigned to applicant
i43	Amount of credit for tuition fees assigned to applicant
i44	Maximum amount of credit for tuition fees
i45	Tuition fees of educational institution of applicant
i46	Requested amount of credit for tuition fees
i47	Tuition fees declared by law
i48	Kind of education of applicant

Table 2.2: Data elements in the student grant example

2.3 TWO ALTERNATIVE DESIGNS

This section presents two workflow process designs for the student grants example. The process designs are based on two different sets of activities, i.e. *activity designs*. Both activity designs are considered *valid* according to [36], as they (i) include all data elements and operations of the PDM, and (ii) respect the dependencies of the informational structure.

2.3.1 *Design 1*

Figure 2.3 presents a process model design for a preliminary set of activities. The process model contains six activities. Their contents are depicted in Figure 2.4. Activity A represents receiving a student's application. It consists of the eleven leaf operations and leaf data elements of the PDM. Activity B then uses the information derived from this application. The activity first determines an applicant's eligibility to receive a grant (i27). After which it continues with the calculation of the amount of basic grant assigned to an applicant (i40). The process model then shows two possible alternatives. Firstly, if i27 is negative, the application is rejected by executing activity H; the workflow process is thereby completed. The second alternative resembles acceptance of the application. In this alternative, the three remaining types of grant are computed by activities C and D; these activities can be executed in parallel. Activity C determines the loan amount (i41) and the amount of tuition fee (i43) that are assigned to an applicant. While activity D determines the assigned amount of supplementary grant (i39). Based on the amounts for the four types of grant, activity G finally computes the total amount assigned to an applicant (i42).

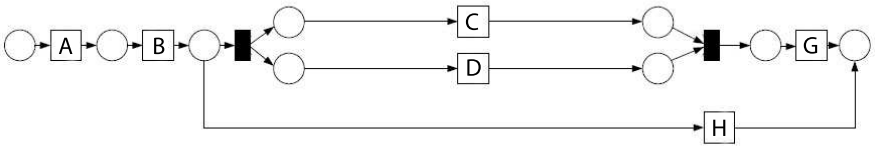


Figure 2.3: Process model of alternative design 1

Design 1 is a valid design that furthermore clearly illustrates distinguishable alternatives for acceptance (activity G) and rejection (H) of an application. Nevertheless, the design has several issues with respect to understandability and execution efficiency. Firstly, activity B can be considered inefficient. i40, the value for the basic grant, is computed even if the application will be rejected based on the value of i27. Furthermore, this activity defers execution of activities C and D until after data element i40 is computed. This is not necessary, since

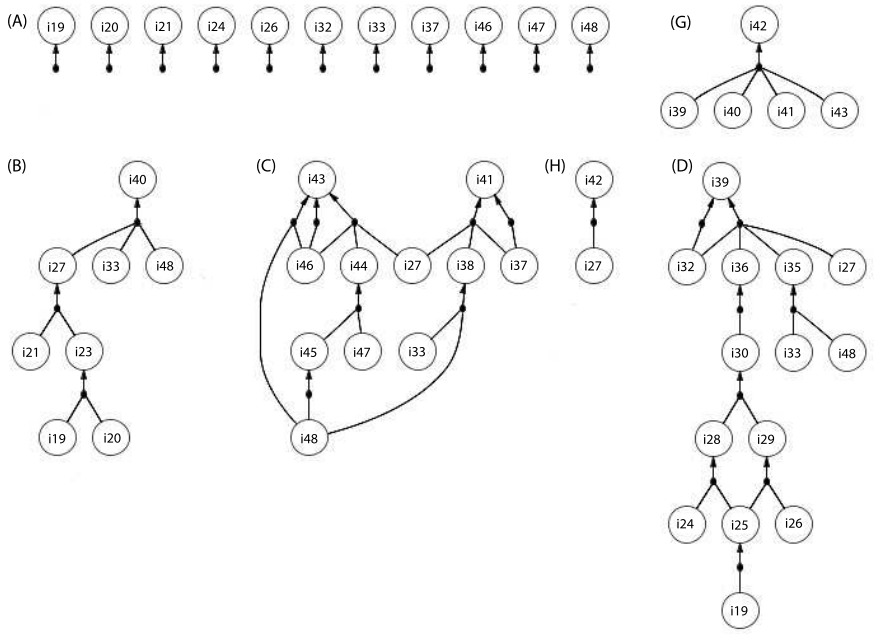


Figure 2.4: Activities of alternative design 1

these activities can be executed independent of element i40. A second issue involves the information that the process model provides about the underlying informational structure. As introduced in Section 2.2, a student grant consists of four types of grant. However, this information cannot be derived from Figure 2.3. Only by analysing the contents of the individual activities, are the four types of grant revealed. The issues that are present in this design can be overcome by applying a small amount of adjustments on the activities. This is illustrated by alternative design 2.

2.3.2 *Design 2*

The second design is reached by making two adjustments to design 1. Activities B and C are both split into two activities, respectively into B1 and B2, and C1 and C2. The resultant process model is shown in Figure 2.5. This process model clearly resembles the core structure of the PDM. The parallelism of activities B2, C1, C2, and D illustrates that the four values (i.e. grants) can be independently computed when an application is accepted. Furthermore, this design overcomes the, earlier mentioned, efficiency issues that are present in design 1. Due to the splitting of activity B, execution of other activities is no longer unnecessarily deferred. As seen in Figure 2.6, activity b1 is completed after i27 has been produced. The final important improvement is related to the meaning of the activities. Activities B and C in design 1 lack a clear focus. Activity B considers eligibility as well as the basis grant, while activity C calculates two unrelated types of

grant. The activities in design 2, by contrast, each have a clear focus and all work towards the production of a single data element. These data elements furthermore represent data elements that are important in the context of the student grant case. Hence, each activity in the second design represents a logical step in the process.

The two alternative process designs illustrate the impact that activity composition can have on execution efficiency, process understandability and meaningfulness of activities. The second design, however, only describes certain characteristics of proper activity designs; it leaves open the issue of how to compose them in a structured way. To address this issue, Chapter 3 first determines which factors influence the quality of an activity design. Afterwards, Chapter 4 defines explicit guidelines to support objective activity composition.

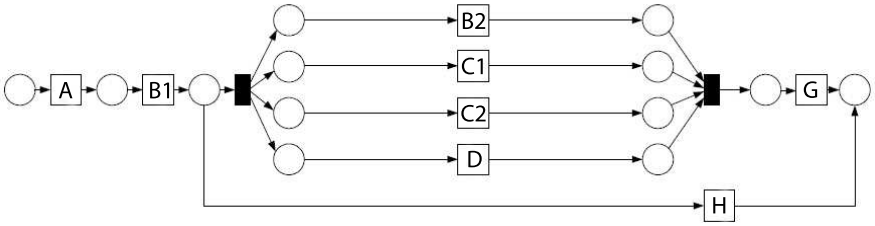


Figure 2.5: Process model of alternative design 2

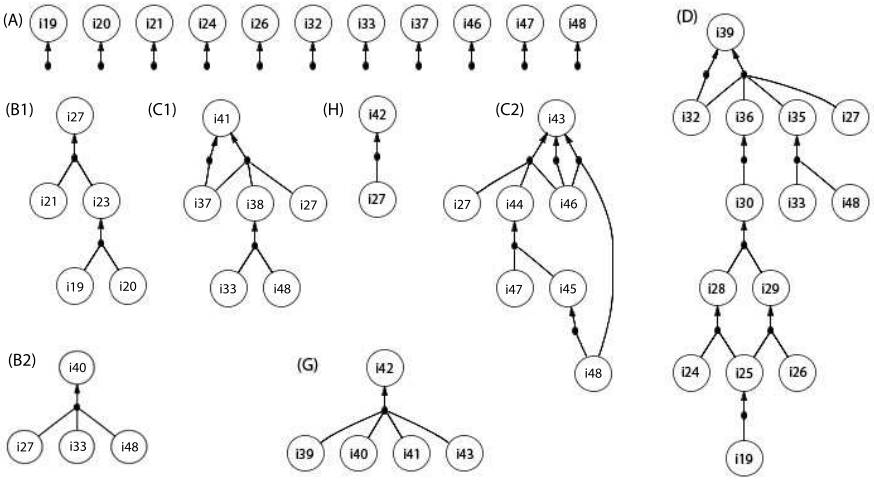


Figure 2.6: Activities of alternative design 2

3

SCOPE

Chapter 3 introduces the factors that can be taken into account during activity composition. The presented factors are those that have been found, most prominently during an extensive literature review. The goal of this chapter is to narrow the scope of the remainder of this research by identifying factors that are *explicit* and *context-independent*. A factor is considered to be explicit if its values can be defined unambiguously, e.g. it is a quantitative factor. Explicitly definable factors are thus suitable to be used in automated activity composition guidelines. Context-independent, here, means that the evaluation of a factor can be performed based on information defined in a PDM. The evaluation should thus not require information about the context of the workflow process. Information that is not expressed in a PDM is, for example, the number of resources that are available to the workflow process. Factors that are explicit and context-independent are considered to be suitable to be incorporated in objective activity composition guidelines.

The presented factors are related to three perspectives. These perspectives each consider one of the three types of advantages that are achievable through activity composition. Section 3.1 presents the *workflow user perspective* to activity composition. It contains a variety of aspects that influence the meaningfulness of individual activities to the workflow users that execute these. The *process execution perspective* of Section 3.2 describes factors that affect the execution efficiency of a workflow. These factors generally encompass multiple activities. Thirdly, the factors that influence *process model understandability* are considered in Section 3.3. This chapter concludes with a recollection of the factors that are explicit and context-independent in Section 3.4. Those factors determine the scope for the remainder of this research.

3.1 WORKFLOW USER PERSPECTIVE

This section considers activity design from the perspective of the workflow user. As introduced in Chapter 2, activities should represent logical units of work to the employees that execute these. The quality of an activity is here considered as a combination of size, i.e. its *granularity*, and meaning, i.e. its *semantics*. Four factors have been found that influence the granularity of an activity: resource requirements (Section 3.1.1), processing time (Section 3.1.2), its representation in a graphical user interface (Section 3.1.3), and complexity (Section 3.1.4). The semantics of an activity are considered based on

FACTOR	EXPLICIT	CONTEXT-INDEPEDENT	INCLUDED
Resource requirements	Yes	Partially	Yes
Processing time	Yes	No	No
GUI content	Yes	No	No
Task complexity	Derivable	No	No
Experienced meaningfulness	Derivable	Yes	Yes
Semantic relatedness	Derivable	Yes	Yes

Table 3.1: Factors in the workflow user perspective

its perceived meaningfulness (Section 3.1.5), and the semantic relatedness of its elements (Section 3.1.6). Table 3.1 shows that three out of these six factors are suitable to be included in objective guidelines.

3.1.1 *Resource Requirements*

The resource dimension of a workflow process considers the means required to execute the process. *Resource* is used as "a generic term for all means that are required to produce a product within the setting of a business process" [24]. In PBWD, a resource class or role can be assigned to an operation. Assigning a resource implies that only a resource of that class is allowed to perform that operation. All the operations in an activity must therefore be associated with the same resource class in order for the activity to be executable.

Evaluation of these resource requirements is explicit and context-independent. Resource requirements are hence suitable to be included in the scope of this research. However, there also may exist a partial ordering between resource classes. A partial ordering means that a resource that is *higher* in the partial ordering, is allowed to perform all operations that can be executed by resources *lower* in the partial ordering [35, p.66]. By considering this ordering during activity composition, it is possible that operations are no longer executed by the resource classes that are originally assigned to execute them. By composing such activities, the workload of higher resource classes increases. This can negatively influence process execution, for example, due to increased utilization of resources (see e.g. [19]). The information required to consider these aspects, such as the number of resources, is not present in a PDM. Consideration of partial ordering is hence not context-independent. For this reason, this thesis abstracts from a (possible) partial ordering in the resource perspective.

3.1.2 *Processing Time*

The processing time of an activity is the amount of time that is required to execute it. This time is directly dependent on the processing times of its underlying operations. When activities are too large, they may become unworkable for workflow users [35]. High processing times may furthermore result in reduced execution flexibility [25]. This implies that the processing time of an activity should not be too large. However, a maximum time depends on the context of the activity. For example, an activity with a processing time of ten minutes can be considered too large for a cashier in a supermarket. By contrast, an activity of the same size may be perfectly acceptable in a workflow that handles insurance claims. For this reason, processing time is considered to be context-dependent. It is therefore not included in the scope of this thesis.

3.1.3 *Graphical User Interface*

The execution of workflow processes is typically supported through the use of information systems [19], most prominently through a Workflow Management System (WFMS). A WFMS defines and manages the execution of workflow processes. It further supports the workflow user by invoking *workflow applications*. These are software programs that support execution of particular activities in a workflow [32]. Figure 3.1 provides an example of an activity that is supported through a workflow application, the figure is derived from [5].

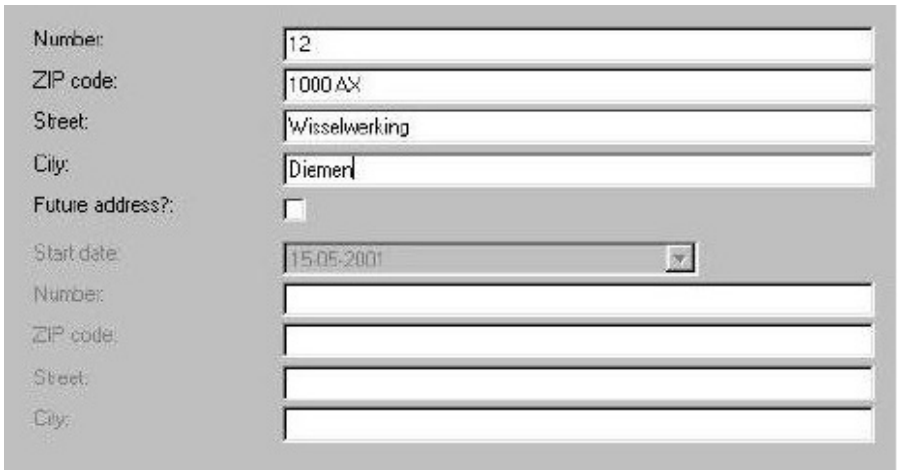


Figure 3.1: Activity supported by a workflow application

Each information field that is shown in the application represents a data element in a PDM. The information fields that are required to be filled in by a workflow user, such as those shown in Figure 3.1, are the data elements that are computed in the activity, i.e. elements computed by executing its operations. The content of an activity, in

the form of its input elements and its operations, thus influences the lay-out of the workflow application, i.e. the Graphical User Interface (GUI). Based on literature from the field of electronic form design, it has been found that the number of information fields, the type of information, as well as their topic, are factors that influence the quality of a GUI. Jarrett [11, p.103], poses that forms do not have a minimal size, as long as they have distinguishable topics. By contrast, Jarrett does state that long forms should be split up. For activity composition, this implies that activities do not have a minimal size, but they should not contain too many operations, or depend on too many data elements. The exact maximum number of elements is, however, dependent on, for example, the GUI lay-out of the workflow application. This information is not expressed in a PDM, and therefore context-dependent. Considering the GUI perspective for activity composition is therefore an aspect that should not be overlooked, but is unsuitable for this research.

3.1.4 *Task Complexity*

The execution of any task or activity adheres to a certain level of complexity, generally referred to as *task complexity*. Many different theories about task complexity exist in literature, such as an *interactionist formulation* that defines complexity in terms of the resources capabilities [4] and Wood's [40] theory of complexity that considers required acts and information cues. In order to incorporate task complexity in this research, it must be made explicit. Buzacott [3] suggests that this may be possible, due an apparent correlation between task complexity and variance in processing time. However, no guidelines have been found that indicate when a task should be considered too complex. It is therefore that the aspect of task complexity is considered to be dependent on the context, and that the design decisions are too subjective. Hence, task complexity is further excluded from this research.

3.1.5 *Experienced Meaningfulness*

Experienced meaningfulness is here considered as the degree to which a workflow user experiences an activity as valuable and worthwhile. *Task identity* and *task significance* are two determinants of this factor in the Job Characteristics Model of Hackman and Oldham [7]. Task identity is the degree to which a job requires completion of a whole and identifiable piece of work, while task significance is the degree to which a job has a substantial impact on the work of other people. The concept of working towards an identifiable goal is also considered as a best practice in electronic form design [11]. These concepts suggest that activities that work towards the production of an important

piece of information, i.e. a data element, are considered meaningful by workflow users. Employees that work on activities that they like are furthermore expected to work more efficiently [26]. In order to compose meaningful workflow activities, important data elements in a PDM must hence be distinguished.

Intuitively, a data element is important due to its semantic meaning. This importance is therefore an implicit property. However, it has been found that semantic importance is also revealed by structural data-flow relations. This implies that it is possible to derive the semantic importance of data elements through the structural relations that are captured in a PDM. Example 3.1 illustrates this relation between semantic and structural importance for the student grants case.

EXAMPLE 3.1 (DATA ELEMENT IMPORTANCE): Data element i27, an applicant's right to receive a grant, is without doubt an important data element in the student grants case. i27 determines whether an application will be rejected or accepted. It is thus a decisive element in the workflow process, and therefore clearly important from a semantic perspective. However, as earlier shown in the PDM of Figure 2.2, data element i27 is required as input for no less than five different operations. The impact of this element on the structural data-flow is hence also eminent. It is through this structural impact that element i27 can also be identified as important, purely based on structural properties.

The presented example is no coincidence; thorough analysis of available cases has revealed that structural properties are generally valid indicators of data element importance. By exploiting the relation between semantics and structure, the importance of data elements can be identified in an objective and context-independent fashion. The design of activities such that they work towards the production of an important data element, as well as the apparent relation between semantic importance and structural properties, form two of the most important principles that are used in the guidelines proposed in Chapter 4. By adhering to these guidelines, the output element of each activity represents an important element of the data-flow.

3.1.6 *Semantic Relatedness*

Semantic relatedness considers the degree to which the meaning of elements is alike. In a PDM, semantic relatedness is most prominent in elements that are related to the same informational product or to a similar topic. An important indicator of semantic relatedness is the description of data elements. For example, consider the five data elements of the student grant example that are presented in Table 3.2. The description of each of the five elements contains the term *tuition fees*. The data elements are thus arguably related through their shared

ID	DESCRIPTION
i43	Amount of tuition credit assigned to applicant
i44	Maximum amount of credit for tuition fees
i45	Tuition fees of educational institution
i46	Applicant has requested credit for tuition fees
i47	Tuition fees declared by law

Table 3.2: Semantic related data elements

relatedness to the topic of tuition fees. From a semantic viewpoint, the core difference between the alternative activity designs of Section 2.3, is based on the relatedness of the elements per activity. The activities in the second, *proper*, design each have a single, clear topic, whereas activities B and C of design 2 were both related to multiple different topics. The operations in the activities of the second design are therefore considered to have better semantic relatedness.

Considering the relatedness of elements based on their descriptions requires natural language analysis that is able to identify semantic relations between elements. This task is often trivial to human modellers, but is far more complex to capture in automated guidelines. Approaches that analyse the labels of process model elements have been proposed [e.g. 13, 14, 15], which indicates that analysing data element descriptions may be challenging, but not impossible. However, as with the aspect of experienced meaningfulness (Section 3.1.5), there once more appears to be a general relation between semantics and structural properties. Figure 3.2, for example, shows that the five data elements of Table 3.2, are all connected in the structure of the PDM. It is proposed that by considering structurally connected operations, semantically related elements of a PDM can be identified. A related proposition is made by the *Cohesion & Coupling* metrics of [36]. These metrics also acknowledge that structural relations in a PDM can be used to evaluate if operations in an activity *belong* together. However, these metrics can only be retrospectively applied on activities that have already been composed. They therefore do not represent design guidelines themselves.

Since structural properties are objective and context-independent, semantic relatedness of elements is suitable to be included in automated design guidelines. This construct presents one of the core principles of the composition guidelines proposed in Chapter 4, together with the notion of data element importance.

3.2 PROCESS EXECUTION PERSPECTIVE

This section considers the effects of activity design on process execution. By grouping operations into activities, five factors in the process

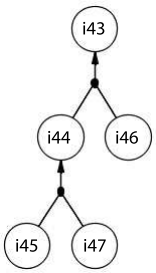


Figure 3.2: Semantic related data elements

FACTOR	EXPLICIT	CONTEXT-INDEPEDENT	INCLUDED
Deadlocks	Yes	Yes	Yes
Defragmentation	Yes	Yes	Yes
Parallelism	Yes	No	No
Deferred execution	Yes	Yes	Yes
External sources	Yes	Yes	Yes

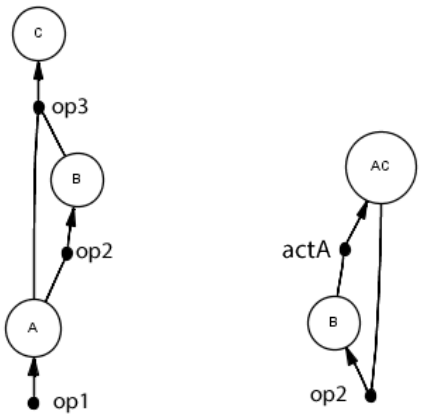
Table 3.3: Factors in the process execution perspective

execution perspective can be considered. The presented factors influence throughput time, as well as execution flexibility. The factors are: deadlocks (Section 3.2.1), defragmentation (Section 3.2.2), parallelism (Section 3.2.3), deferred execution (Section 3.2.4), and external information sources (Section 3.2.5). Table 3.3 shows that only parallelism is excluded from the research scope. This factor cannot be evaluated in a context-independent manner.

3.2.1 *Deadlocks*

When combining operations, the least desirable result is that the process is no longer executable. Non-executable processes occur when the aggregation of operations results in cyclic behaviour. Composition of activities that result in deadlocks should hence, straightforwardly, be avoided. Example 3.2 illustrates how deadlocks can occur.

EXAMPLE 3.2 (DEADLOCKS): Consider composition of an activity actA, that consists of op1 and op3 in the PDM of Figure 3.3a. op1 computes a value for a data element (B) that is required by operation op2, while op3 actually depends on a value that is produced by that operation (op2). By aggregating op1 and op3 into actA, a situation occurs where op2 can only be executed after actA *and* vice versa. This results in a non-executable construct, i.e. a deadlock. Such a construct is visualized as a cycle, as shown in Figure 3.3b.



(a) Before aggregation (b) After aggregation

Figure 3.3: Occurrence of a deadlock

3.2.2 *Defragmentation*

Any grouping of operations into activities reduces the number of model elements, i.e. results in defragmentation of the model. Defragmentation of tasks decreases the number of hand-overs between employees. Fewer hand-overs lead to reductions in set-up time [1] and coordination effort [25, 41]. Seidmann and Sundararajan [29] further pose that there is a possibility of error that arises with every hand-off. A reduction in the number of hand-overs is hence desirable. It should however be noted that aggregation of operations is a way to enforce such a reduction in a relatively inflexible manner. Work item interaction strategies, such as those proposed in [9], can be applied to achieve similar results, while preserving a higher level of flexibility. As work item strategies are beyond the scope of the PDM, defragmentation of workflows is here regarded as a beneficial aspect of activity composition.

3.2.3 *Parallelism*

In business processes, tasks can be concurrently executed in order to reduce the throughput time of a case. As an activity is executed by a single resource, only one operation in an activity can be executed at once. When parallel operations are combined into an activity, this thus diminishes the possibility of simultaneous execution. Simplified calculations, such as presented in [20], imply that throughput time is reduced when replacing a sequential by a parallel design. Other authors [e.g. 1, 25, 41], however acknowledge that parallel processes are more complex than sequential processes and thus also have drawbacks. The additional complexity of parallel processes indicates that it is often beneficial, with respect to resource utilization and average throughput time, to aggregate parallel operations due to reductions

in hand-overs and coordination effort [41]. Coordination effort and resource utilization are both aspects that are not expressed in a PDM, i.e. they are context-dependent. It is therefore not possible to determine whether a reduction in parallelism is beneficial for a given PDM. In the remainder of this research, concurrency is thus neither purposely reduced nor preserved.

3.2.4 *Deferred Execution*

The possibility to start execution of activities may be delayed due to aggregation of operations. Two cases can be distinguished that result in such a loss of flexibility. Firstly, by increasing the number of operations in an activity, it is possible that the set of input elements of an activity is increased. Execution of an activity can only commence when all required input elements are available, hence execution may be deferred due to these increased requirements.

The second case in which the execution of activities is deferred, occurs when activities continue execution after a data element that is required for other activities has been produced. An example of this is seen in design 1 of Section 2.3.1. There, activity B continues execution after data element i27 has been computed. This defers execution of those activities that require i27 as input, i.e. activities C, D and H.

The former case of deferred execution only defers (part of) a single activity. Its impact is thus limited. The negative impact of the second case is, however, larger. Such constructs can defer the execution of multiple activities. Furthermore, as illustrated by the alternative design of Section 2.3.2, this construct can be avoided by correctly splitting activities. It is therefore posed that activity composition should avoid such deferred execution.

3.2.5 *External Sources*

Aside from *regular operations*, which operate on a set of input elements and produce a value for an output element, a second type of operations exists in PDMs: *leaf operations*. A leaf operation has no input elements and produces a value for a *leaf element*. Leaf elements are those information elements that are provided as inputs to the process. Their values can be retrieved from three sources: (i) internally from the same institution, (ii) from the client, and (iii) from external parties [37].

In practice, multiple leaf elements are often retrieved from the same information source, e.g. in the student grant example, all data elements are retrieved from the student's application. The leaf operations that produce these elements thus share an information source, they can hence be considered to be semantically related. From a semantic perspective, it therefore seems valid to group all leaf opera-

FACTOR	EXPLICIT	CONTEXT-INDEPEDENT	INCLUDED
Labelling	Yes	Yes	Yes
Complex constructs	Yes	Yes	Yes

Table 3.4: Factors related to understandability

tions that are related to a single source into one activity. Activities that consist of grouped leaf operations are found in [e.g. 23, 35, 36]. Such activities resemble events in which the information is received from an external source, e.g. receiving an application in the student grant example. By composing *leaf activities*, the process retrieves multiple information elements from a source at the same time. This adheres to the *contact reduction* best practice from [25]. This practice advises to "reduce the number of contact with customers and third parties".

By considering information sources, it is thus possible to create meaningful activities that also result in improved execution efficiency. The notion of information sources can be added in a straightforward manner to the definition of a PDM. Therefore, the consideration of information sources is explicit and context-independent. Hence, this factor is included in the guidelines presented in Chapter 4.

3.3 PROCESS MODEL UNDERSTANDABILITY

It has been found in [30] that the most prominent use case for BPMA is to gain a *quick overview* of a process. This implies presenting a process model that contains coarse-grained tasks, in which semantically related elements are grouped together. It is exactly this purpose that can be achieved through activity composition. This section therefore considers how activity composition can be used to create understandable process models. Two aspects of understandability are considered: labelling of aggregated elements (Section 3.3.1) and avoidance of complex constructs (Section 3.3.2). Table 3.4 shows that both aspects are suitable for the scope of this research.

3.3.1 *Labelling*

During activity composition, operations are aggregated into activities, an act that results in process defragmentation. Defragmentation is desirable from the perspective of understandability, because larger models tend to be more difficult to understand than smaller models [16, 17]. An issue that arises due to such aggregations, is how these aggregated elements must be labelled [30]. Leopold et al. [14] address this issue, by introducing five naming strategies for aggregated elements in process models. Their *End Event* strategy appears most

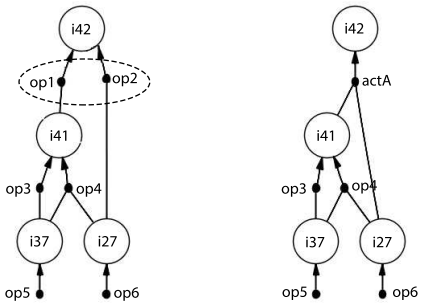
applicable for the activities designed in this paper. The authors pose that "when the state [...] at the end of the process defines the overall goal of the process, the name of the whole model may be closely related [...]." This notion is highly suitable here, because meaningful activities, as defined in Section 3.1.5, have important data elements as their output elements. Therefore, aggregated data elements shall be labelled according to the output element of the activity that produces them. This labelling strategy consequently results in the omission of insignificant data elements, while the important elements are retained in process models. This is essentially the goal of BPMA, as described in [e.g. 21, 31].

3.3.2 *Complex Constructs*

By aggregating operations, it is possible that complex routing constructs may occur in the resultant process models. One such example is the cyclic construct described in Section 3.2.1. A different control flow pattern that can result from activity composition is the *OR-join*. Informally, an OR-join is a construct that is enabled when a subset of its input elements are available. The constructs thus combine properties of XOR- and AND-joins. OR-joins can occur because workflow processes may contain alternative paths to produce a value for the desired end-product. This is for example seen in the student grant case, where there are alternative paths for acceptance and rejection of an application. When alternative paths are combined into a single activity, this can lead to the occurrence of OR-joins. These constructs appear when an activity contains alternative operations that do not require the same input elements. This notion shall be referred to as *redundant input* of an activity. Example 3.3 illustrates how activities with redundant input result in OR-constructs.

EXAMPLE 3.3 (REDUNDANT INPUT): Figure 3.4a presents a fragment of the student grant example. It contains four data elements and six operations. Consider a possible aggregation of the alternative operations that produce i42: operations op1 and op2. It is clear from Figure 3.4a that these alternative operations require different input elements. By aggregating these two operations, an activity with redundant input is created. The proposed aggregation is depicted in Fig. 3.4b. The activity, actA, requires the computation of input elements i27 and i41. However, as seen in Fig. 3.4a, the computation of i41 is not required when op2 is executed. Therefore, actA requires either one, or two input elements. It thus represents an OR-construct, even though it is visualized as an AND-join.

OR-join constructs have been found to result in process models that are hard to understand and are ambiguous [16]. Furthermore, as illustrated in Example 3.3, these constructs hide important infor-



(a) Before aggregation (b) After aggregation

Figure 3.4: Activity with redundant input

mation from a process model. The aggregated PDM of Figure 3.4b does not show that applications can be accepted *and* rejected, while this is arguably important information. Despite these issues, literature on manual activity composition does not always adhere to this restriction. An occurrence of a loss of important information found in literature is illustrated in Example 3.4.

EXAMPLE 3.4 (REDUNDANT INPUT IN LITERATURE): Figure 3.5 depicts a fragment of a process model shown in [35, p.423]. The fragment is part of a manual activity design for the student grant example. The two activities in the fragment are depicted in Figure 3.6. Activity E has two possible execution alternatives. One of these alternatives requires a value for data element i30, which is produced by activity A. The other alternative operation, however, does not require any data element that is produced by activity A. Therefore, activity E does not necessarily require execution of activity A. Hence, activity E is said to have redundant input elements. Although the presence of two alternative operations impacts the execution of other activities, this information cannot be derived from the process model, but is only revealed when the individual activities are inspected.

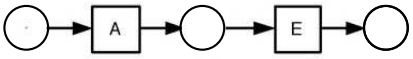
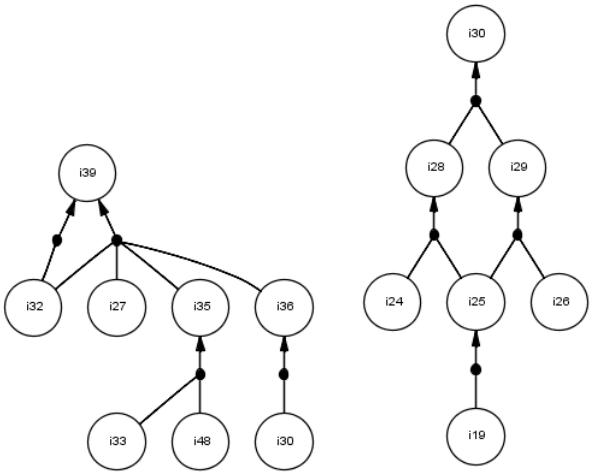


Figure 3.5: Redundant input in literature – process model fragment

Regardless of occurrences in literature, process models with such ambiguous structure are here considered to hide important information. Hence, activities with redundant input elements are prohibited in this research.

3.4 SELECTION

The previous sections have shown that a large number of factors can be considered when composing activities. It has furthermore been



(a) Activity A (b) Activity E

Figure 3.6: Redundant input in literature – activities

found that not all these factors are suitable to be incorporated in objective composition guidelines.

Table 3.1 shows that three out of the six factors of the workflow user perspective are suitable for this research. *Experienced meaningfulness* and *semantic relatedness* are both factors that can be derived from the structural data-flow relations of a process. The former poses that activities should work towards an important data element, while the latter implies that activities should consist of semantically related operations. These notions form the core of the composition guidelines that are presented in Chapter 4. The third suitable factor of the workflow user perspective is the *resource aspect*. However, since this factor depends on information that is not available for every PDM, it is less prominent in the proposed guidelines. The remainder of this research abstracts from the factors that have been found to be context-dependent: processing time, GUI content, and task complexity.

Section 3.2 considered five factors that influence the execution efficiency of a workflow. Most importantly, this section showed that activity composition must not lead to *deadlocks*. Further, it has been illustrated how *deferred execution* can occur, and hence, how it can be avoided through proper activity composition. Finally, by considering the *external sources* from which leaf elements are derived, meaningful leaf activities can be created that further improve process execution. Parallelism is the single factor of this perspective that has been found to be unsuitable for inclusion in the scope, because it depends on information about the context of a workflow.

Lastly, Section 3.3 showed that the proper *labelling of elements*, and the *avoidance of complex constructs* improve the understandability of workflow processes. These factors are therefore also considered when composing activities.

Chapter 4 presents composition guidelines that comprise these selected factors. The guidelines emphasise the factors that can be derived from structural properties, such as meaningfulness, semantic relatedness and deferred execution. Resource requirements and external information sources are then used to refine activities, when information beyond structural data-flow relations is available.

4

COMPOSITION GUIDELINES

This chapter presents guidelines to support activity composition for information-intensive workflow processes. The presented guidelines are based on those factors that have been found in Chapter 3 to be *explicit* and *context-independent*. The guidelines can therefore be applied in an automated fashion and do not require information beyond the specification of a PDM. The goal of these guidelines is to approximate design choices made by experienced modellers.

The most important guidelines pose that activities should work towards production of an important data element and consist of operations that are semantically related to each other. It is proposed that this can be achieved based on a minimal amount of available information: the structure of a PDM. To objectify this guideline, Section 4.1 considers five structural patterns that are proposed to reveal important data elements. Section 4.2 then illustrates how those important data elements can be used to compose activities that consist of semantically related operations. Finally, Section 4.3 presents additional guidelines that can be used to further refine activities when additional process information is available.

4.1 DATA ELEMENT IMPORTANCE

Recall from Section 3.1.5, that the meaningfulness experienced by workflow users during activity execution can be increased by designing activities that complete an identifiable piece of work. It is therefore that activities shall be designed such that the final data element that is produced, i.e. the *output element*, is an important element in the informational structure. In order to achieve this goal, this section sets out to identify such important data elements. Section 3.1.5 introduced the relation between the semantic importance of data elements and their structural properties. This section proposes that there are five structural patterns in a PDM that reveal this relation. These patterns have been found based on intuition, experimentation, and by analysing manual activity designs in [e.g. 24, 35, 36]. The identified elements are referred to as *important data elements*. The proposed structural patterns are motivated by referring to the student grants example of Section 2.2. However, it is proposed that these patterns are applicable to PDMs in general. This proposition is later evaluated in Chapter 6.

The five types of important data elements are: (i) the root data element (4.1.1), (ii) leaf data elements (4.1.2), (iii) conditional data el-

ements (4.1.3), (iv) equal-level data elements (4.1.4), and (v) reference data elements (4.1.5).

4.1.1 *Root Data Element.*

The root element is clearly the most important data element in a PDM, because this data element represents the desired final outcome of a workflow. For example in the student grants case, this is the total amount of student grant assigned to an applicant (i42). Pattern 1 therefore simply identifies the root data element.

PATTERN 1 (ROOT DATA): The root data element is the single data element that is not used as input for any operations.

4.1.2 *Leaf Data Elements.*

The second type of important data elements that can be structurally distinguished are leaf elements. Leaf elements are data elements produced by leaf operations. These are the information elements that are provided as input to a process; the values for these elements are retrieved from outside the process. In the student grant example, the leaf elements represent the information that is retrieved from a student's application, e.g. i20, the birth date, and i21, the nationality of the applicant.

PATTERN 2 (LEAF DATA): A leaf data element is a data element that is produced by an operation without input elements.

4.1.3 *Conditional Data Elements*

The third type of important elements revealed by structural patterns are *conditional data elements*. Conditional data elements are those data elements that can be produced by multiple, alternative operations. Four such elements exist in the student grant example: the root element i42, and elements i39, i41, and i43. These latter data elements represent three out of the four types of grant that applicants may be eligible to receive. These grants directly affect the value of the root element of the PDM and are therefore clearly important in this process.

PATTERN 3 (CONDITIONAL DATA): A conditional data element is a data element that can be produced by multiple alternative operations.

4.1.4 *Equal-Level Data Elements*

By considering conditional data, the importance of three out of the four types of grant, in the student grant example, has been identified. The fourth type of grant is the amount of basic grant assigned to an

applicant, represented by i40. Without additional information, there is no reason why this fourth grant should be considered as less important than the other types of grant. i40 is hence also considered to be of importance. The similarity between i40 and the other three types of grants is also revealed through their structural relations. Figure 4.1 shows that the operation that produces i42, requires all four types of grants as input elements. It is therefore proposed, that since i40 is required as input for an operation that requires other important data elements, i.e. conditional data elements, i40 is important as well. It is proposed that this transitive notion can be applied in a generic fashion to uncover the fourth type of important data elements: *equal-level data elements*. These are described in Pattern 4.

PATTERN 4 (EQUAL-LEVEL DATA): An equal-level data element is an input data element to an operation that also requires conditional data as input.

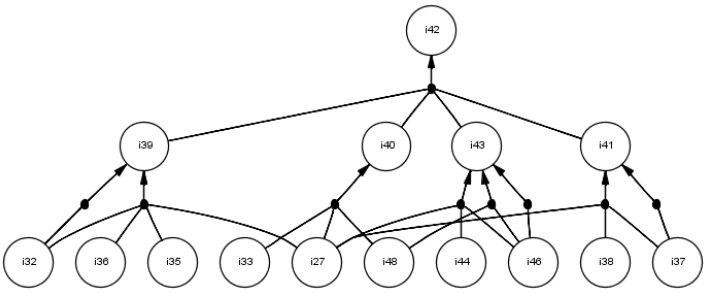


Figure 4.1: Example of equal-level data

4.1.5 *Reference Data Elements*

Recall from Section 3.1.5, that i27, the eligibility to receive a grant, is arguably an important data element in the student grant example. i27 is important from a structural perspective, because the data element is required for the execution of operations that produce multiple important data elements. Such data elements shall be referred to as *reference data elements*. Reference data elements represent the final type of important data elements. The identification of reference data elements as important data elements is furthermore beneficial for execution efficiency. This ensures that activities do not unnecessarily defer execution of multiple other activities. They also result in a strict avoidance of deadlocks.

PATTERN 5 (REFERENCE DATA): A reference data element is a data element that is an input element to multiple operations that are, directly or indirectly, involved in the computation of different sets of important data elements.

4.1.6 *Proposition*

The five patterns represent those structural data-flow relations that are proposed to predict the significance of data elements. This results in the first proposition for objective activity composition, Proposition 1.

PROPOSITION 1 (IMPORTANT DATA ELEMENTS): Root, leaf, conditional, equal-level and reference data elements represent important data elements in a PDM.

4.2 SEMANTIC RELATEDNESS

Semantic relatedness considers the degree to which the meaning of elements is alike. As introduced in Section 3.1.6, it is here proposed that operations with similar meaning can, to a certain extent, be identified based on structural properties of a PDM. The underlying intuition is that each operation can be associated with the computation of a single important data element. Recall that these elements are identified by the five structural patterns considered in Section 4.1. Operations that are associated with the same important data element are then considered to be semantically related. Definition 4.1 first defines how operations are associated with important data elements.

DEFINITION 4.1 (ASSOCIATED DATA ELEMENT): An important data element is associated with an operation, if there exists a path in the PDM from the operation to that data element, such that this path does not contain any other important data elements.

Due to the way important data elements, especially reference data elements are defined in Section 4.1, each operation is associated with exactly one important data element. This is captured in Lemma 1. The proof for this lemma is provided in Appendix A.

LEMMA 1 (UNIQUE ASSOCIATED DATA ELEMENT): Each operation has exactly one associated data element.

As operations that are associated to the same data element are considered to be semantically related, it is proposed that such operations can be grouped into a semantically coherent activity. Composition of these activities should however not lead to activities that have redundant input elements. The notion of input redundancy has been introduced in Section 3.3.2. Activities with redundant input are prohibited, because these require the use of complex constructs, hide routing information from process models, and can possibly result in the computation of superfluous data elements. It is therefore that activity composition is restricted by Constraint 1.

CONSTRAINT 1 (REDUNDANT INPUT): An activity may not be designed such that it would be possible to execute the activity without first computing all of its input elements.

It is proposed that semantically coherent activities can be composed by considering associated data elements and prohibiting composition of activities with redundant input elements. This proposition is extended with the notion that leaf operations can be grouped together into *leaf activities*. This idea is introduced in Section 3.2.5, and also found in manual activity designs in [e.g. 35, 36, 24]. This notion is based on the premise that the values for multiple leaf elements are often retrieved from the same information source. For example, as seen in Sect. 2.2, all leaf elements in the student grants case are derived from a student's application. By grouping leaf operations, a process retrieves multiple information elements at once. This reduces the number of contacts with customers and third parties, which is desirable according to [25]. Section 4.3 considers how information sources can be utilized to further refine leaf activities.

4.2.1 *Propositions*

Proposition 2 summarizes the ideas presented in Section 4.2.

PROPOSITION 2 (SEMANTICALLY COHERENT ACTIVITIES): A semantically coherent activity is an activity that consists of a set of operations that are associated with the same data element and of which the aggregation does not result in redundant input.

It is finally proposed that activities which adhere to Proposition 1 and Proposition 2, are well-designed activities. These activities work towards a clear goal, and consist of semantically related operations. This is defined in Proposition 3.

PROPOSITION 3 (WELL-DESIGNED ACTIVITIES): The operations in an activity should work towards the production of a relatively important data element and be semantically related to each other.

By applying Proposition 3, it is possible to automatically generate a set of activities based on the structure of a PDM. Activity designs that adhere to this composition guideline furthermore ensure good execution efficiency, since these designs avoid deadlocks and deferred activity execution. The second alternative of Section 2.3 presents an example of an activity design that is generated based on these propositions.

When additional process information is available, this can be used to further refine the generated activities. This refinement is considered in Section 4.3. Chapter 5 presents an implementation of the proposed guidelines. These are furthermore quantitatively evaluated in Chapter 6.

4.3 REFINEMENT

As introduced in Section 3.4, the resource perspective and external information sources are two factors that can also be included in objective composition guidelines. This section considers the refinement of activities that have been generated based on Proposition 3. While the previously considered guidelines can be applied for any PDM, the guidelines that are presented here, require the availability of information beyond structural relations. Section 4.3.1 presents the refinement of activities by including resource requirements, whereas Section 4.3.2 illustrates how information sources can be used to improve the quality of leaf activities.

4.3.1 *Resource Requirements*

Work on activity design in PBWD [e.g. 26, 36], acknowledges that there must exist at least a single resource class that can execute all operations in an activity. These resource requirements have been introduced earlier in Section 3.1.1. Information about resource requirements can be incorporated in the earlier propositions in order to ensure that an activity can always be executed by a single resource. This is achieved by introducing a sixth pattern that reveals an important data element: *hand-over data elements*.

PATTERN 6 (HAND-OVER DATA): A hand-over data element is the output element of an operation that is executed by a particular resource class, while the data element is also an input element for an operation that is executed by a different resource class.

By extending the set of important data elements with hand-over data, Proposition 2 ensures that an activity never requires multiple resource classes. Unfortunately, the resource perspective has not received a lot of attention in research related to PBWD [35, p.228]. It is therefore that there are no examples of PDMs with detailed, and explicit, resource specifications available. An example introduced in [6], however, considers a bicycle manufacturing process that requires involvement of different departments. Example 4.1 considers the refinement of an activity in the bicycle process based on supposed resource requirements, in order to illustrate the benefits that can be gained by including resource specifications.

EXAMPLE 4.1 (REFINEMENT THROUGH RESOURCES): Figure 4.2 presents an activity that is generated for the bicycle case of [6, p.78]. Descriptions of the data elements are provided in Figure 4.1. The activity works towards the assembly of a bike, represented in i24. The activity appears to comprise two distinct parts. Firstly, the two operations that produce i24 are alternative operations that represent the assembly of a bike. All other operations are related to a back-ordering

process, as can be derived from the data element descriptions in Table 4.1. This back-ordering process is executed when not all required parts are available, i.e. i15 is negative.

For illustrative purposes, it is assumed that the two operations that produce i24 are performed by a resource from the manufacturing department (R_M). Whereas the operations related to the back-ordering of parts are performed by a resource from the purchasing department (R_P). Data element i23 is produced by an operation that is executed by resource R_P, while it is required for an operation that is performed by R_M. For this reason, i23 is a hand-over element.

After the identification of hand-over elements, refined activities can be generated based on Proposition 2. The resultant process model contains three activities, as depicted in Figure 4.3. These activities are required, because a division into two activities would result in an activity with redundant input elements. (Example 3.3 illustrates this for a very similar case). The process model presents two clear alternatives: if all required parts are available (i15 is positive), the bike is assembled by activity A. If i15 is negative, the missing parts are back-ordered through execution of activity B. After the back-ordered parts have been received, activity C assembles the bike. This process model design arguably reveals important details which were previously hidden in the activity design without resource specifications, i.e. the existence of two alternatives, as well as a distinguishable back-ordering process,

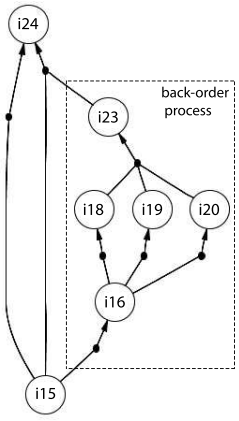


Figure 4.2: Activity in bicycle case

ID	DESCRIPTION
i15	Availability of parts (yes/no)
i16	Required backordered parts
i18	Payment terms of vendors
i19	Quality guarantees of vendors
i20	Delivery options of vendors
i23	Backordered parts
i24	Assembled Bike

Table 4.1: Data elements in bicycle activity

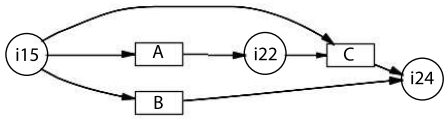


Figure 4.3: Process model of bicycle fragment

Example 4.1 has illustrated that the inclusion of a resource specification can result in an activity design that better resembles the workflow process. It is for this reason, that the resource perspective is utilized whenever this information is available.

4.3.2 *Information Sources*

Recall from Section 4.2 that leaf operations are considered semantically related because they commonly represent retrieving information from a similar source. It is therefore that such leaf operations are grouped together into a leaf activity. If no information about information sources is available, it is assumed that all leaf operations are related to the same information source. This is, for example, true in the student grants case, where all leaf elements are retrieved from a student's application. However, workflow processes may also retrieve information from several sources. An occurrence of this is found in the PDM of the *unemployment* case, considered in [24, p.203]. This example considers a workflow process of a governmental agency. While handling applications for unemployment benefits, the agency uses information derived from two different sources. Some relevant information is stored in the information systems of the agency, whereas other information is retrieved from a client's application. This separation of information sources is reflected in the workflow process design of [24, p.216]. This workflow process contains two leaf activities: one for information retrieved from the agency's internal systems, and one for the information retrieved from a client's application. By incorporating information about the sources from which leaf data elements are derived, it is thus possible to refine the quality of leaf activities. Although information sources are not currently part of the formal definition of a PDM (see [35, p.65]), including this notion is trivial.

This chapter has introduced fundamental guidelines for the objective generation of activities based on the structure of a PDM. Sections 4.3.1 and 4.3.2 have further shown that these activities can be refined if additional process information is available. The proposed guidelines are implemented as a proof of concept in Chapter 5. The quality of activities that are generated by these guidelines is quantitatively evaluated in Chapter 6.

5 IMPLEMENTATION

This chapter presents an implementation of the design guidelines of Chapter 4. The implementation generates activities that have a single important data element as output, and of which the operations are semantically related. These activities can further be refined if information is available about resource requirements and external information sources.

Section 5.1 describes an algorithm that results in activities that adhere to the proposed guidelines. Secondly, Section 5.2 describes freely available software that supports this implementation. The software furthermore enables users to incorporate their personal preferences in the generated activity designs.

5.1 ALGORITHM

This section describes an algorithm that implements the composition guidelines. The implementation is intended as proof of concept that shows that the guidelines can be automated. For this reason, the presented algorithm emphasises understandability, rather than efficiency. Furthermore, to preserve readability, the implementation of certain concepts is simplified in this section. In those cases, Appendix B presents more complex and thorough alternatives. The algorithm consists of four subsequent steps:

- STEP 1: identify the root, leaf, conditional and equal-level data elements (Section 5.1.1);
- STEP 2: use the results of Step 1 to identify reference data elements (Section 5.1.2);
- STEP 3: compute the sets of operations that are associated with the elements identified in the previous steps (Section 5.1.3);
- STEP 4: split those sets, resultant from Step 3, that would lead to activities with redundant input (Section 5.1.4).

5.1.1 *Step 1: Identifying Four Patterns*

The algorithm first identifies the root, leaf, conditional and equal-level data elements. Identification of these four types of goals on the basis of the structure of a PDM is fairly straightforward:

- ROOT DATA ELEMENT. The root data element is the single data element that is not required as input for any operation;

- CONDITIONAL DATA ELEMENTS. Conditional data elements are the data elements that are produced by multiple operations;
- EQUAL-LEVEL DATA ELEMENTS. Equal-level data elements are identified by considering all operations that require a conditional data element as input. All data elements, that are input elements of those operations, represent equal-level data elements. This concept is illustrated by Figure 4.1;
- LEAF DATA ELEMENTS. Leaf data elements are identified as those elements that are produced by leaf operations, i.e. operations without input elements.

Algorithm 1 depicts how these important data elements can be formally identified. Note that the notation uses *predecessors(x)* and *successors(x)* to denote the elements that, respectively, precede and succeed an element x in the PDM.

Algorithm 1 Identification of important data elements

```
1:  Input: PDM
2:  result = ∅
3:  for d ∈ PDM.dataElements do
4:      if successors(d) == ∅ then
5:          result.add(d)                          ▷ d is the root element
6:      if size(predecessors(d)) > 1 then
7:          result.add(d)                  ▷ d is a conditional data element
8:          for op ∈ successors(d) do
9:              result = result ∪ predecessors(op)  ▷ equal-level elements
10:     for op ∈ predecessors(d) do
11:         if predecessors(op) == ∅ then
12:             result.add(d)                      ▷ d is a leaf element
13: return result
```

5.1.2 *Step 2: Identifying Reference Data Elements*

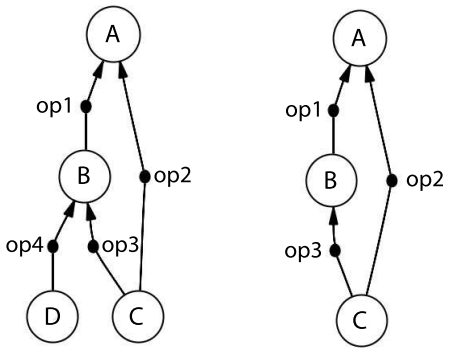
Pattern 5 defines a reference data element as a data element that is an input element to multiple operations that are, directly or indirectly, involved in the computation of different sets of important data elements. This definition reveals that the identification of reference data elements is less straightforward than the identification of the other patterns, as described in Section 5.1.1. Firstly, this is because the identification of reference data requires the other patterns to have been previously identified. Furthermore, the definition reveals that indirect involvement between elements must also be considered. An operation is said to be, directly or indirectly, involved in the computation of a data element, when there exists a path from the operation to that

data element. This notion is formalized, for any PDM elements, in Definition 5.1.

DEFINITION 5.1 (INVOLVEMENT): An element *el1* is involved in the computation of an element *el2*, when there is a path from *el1* to *el2* in the PDM.

These involvement-relations are identified by computing the *transitive closure* of a PDM. A transitive closure G^* contains all pairs (i, j) for which there is a directed path from i to j in a graph [12]. Based on Definition 5.1, a reference data element is a data element for which its succeeding operations are involved in the computation of different sets of important elements. Examples 5.1 and 5.2 illustrate the identification of reference data elements based on involvement-relations.

EXAMPLE 5.1 (REFERENCE DATA ELEMENTS 1): Consider the example PDM of Figure 5.1a. Data element C may be a reference data element, because it is used as input by multiple operations: op1 and op2. To determine if this is indeed the case, the involvement-relations for op1 and op2 are computed. op1 is involved with the conditional data elements A and B; op2 is only involved with data element A. Therefore, the operations that require data element C as input, are involved with differing sets of important data elements. Hence, data element C is identified as a reference data element.



(a) Reference data (b) No reference data

Figure 5.1: Identifying reference data elements

EXAMPLE 5.2 (REFERENCE DATA ELEMENTS 2): Consider the PDM depicted in Figure 5.1b. It is similar to Figure 5.1a, aside from a single, crucial difference: data element B is no longer a conditional data element. Due to this alteration, op1 is now only involved with the computation of one important data element (element A). As the involvement of op2 remains unchanged, op1 and op2 are now involved with identical sets of important data elements. Hence, in Figure 5.1b, data element C is not a reference data element.

Algorithm 2 implements the identification of reference data elements. It takes the important data elements that are identified by

Algorithm 1 as input. The notation uses *involvedWith(x)* to denote the set of elements with whose computation element x is involved. After completion of Algorithm 2, all important data elements have been identified.

Algorithm 2 Identification of reference data elements

```
1:  Input1: PDM
2:  Input2: goals                ▷ Data elements identified by Algorithm 1
3:  result = ∅
4:  for d ∈ PDM.dataElements do
5:      for op1 ∈ successors(d) do
6:          op1Goals = involvedWith(op1) ∩ goals
7:          for op2 ∈ successors(d) do
8:              op2Goals = involvedWith(op2) ∩ goals
9:              if op1Goals ≠ op2Goals then
10:                 result.add(d)
11: return result
```

5.1.3 *Step 3: Computing Associated Operations*

When the important data elements have been identified, it is possible to compose activities that compute these. This first requires the identification of the operations that are associated with these elements. Recall from Definition 4.1 that the associated data element of an operation is the single important data element that can be reached via a path that does not contain any other important data elements. To identify the operations that are associated with a particular element, this definition is applied inversely. First, *prerequisites* are defined in Definition 5.2.

DEFINITION 5.2 (PREREQUISITES): The prerequisites of an element el1 are all elements el2, for which there is a path from el2 to el1 in the PDM.

A prerequisite is the inverse of the involvement-relation described in Section 5.1.2. Identification of prerequisites is therefore similar. They are identified by (temporarily) reversing all arcs in the PDM, and afterwards again computing the transitive closure G^*. The operations that are associated with a data element d1 are then computed by taking the prerequisite operations of d1, and removing the prerequisites of all important data elements d2, for which there exists a path from d2 to d1, i.e. G^* contains $(d2, d1)$. Example 5.3 illustrates this for a fragment of the student grant case.

EXAMPLE 5.3 (ASSOCIATED OPERATIONS): Consider the fragment of the student grant case shown in Figure 5.2. It contains the conditional data elements i41 and i42. Table 5.1 presents the prerequisites

of these elements; they are also marked in the figure. The associated operations of i42 are identified by removing the prerequisites of i41 from the perquisites of i42. This subtraction is required because there exists a path from i41 to i42. This is also identifiable based on prerequisites, since i41 is part of the prerequisites of i42. The associated operations of i42 are then {op1, op2}, while {op3, op4, op5, op6} are associated with i41.

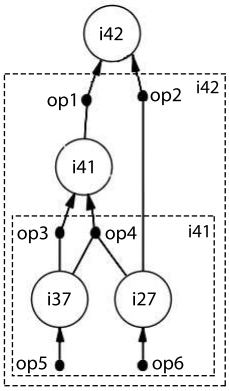


Figure 5.2: Identifying associated operations

ID	PREREQUISITES
i41	{op3, op4, op5, op6, i27, i37}
i42	{op1, op2, op3, op4, op5, op6, i27, i37, i41}
i42 - i41	{op1, op2, i41}

Table 5.1: Prerequisite elements

Algorithm 3 implements the computation of associated operations. The algorithm takes a set of important data elements as input. For each of these elements, it computes the set of its associated operations. Step 3 concludes by grouping all leaf operations together into a leaf activity. The final step in the composition of activities is to avoid sets of operations that result in activities with redundant input. This step is described in Section 5.1.4.

Algorithm 3 Computing associated operations

```
1: Input1: PDM
2: Input2: goals          ▷ Data elements identified by Algorithms 1 and 2
3: activities = {}
4: for d1 ∈ goals do
5:     activity = prerequisites(d1)
6:     for d2 ∈ goals do
7:         if d1 ≠ d2 and d1 ∉ prerequisites(d2) then
8:             activity = activity − prerequisites(d2)
       activities.add(activity)
9: return result
```

5.1.4 *Step 4: Avoiding Redundant Input*

Proposition 2 states that activities should not have redundant input elements. This occurs when alternative operations in a (proposed)

ID	PREREQUISITES
op1	{op3, op4, op5, op6, i27, i37, i41}
op2	{op6, i27}
op1 − op2	{op3, op4, op5, i37 , i41}

Table 5.2: Prerequisites revealing redundant input

activity require the computation of different sets of important data elements. This can be identified by analysis prerequisites, as seen in Section 5.1.3. Example 5.4 illustrates this.

EXAMPLE 5.4 (REDUNDANT INPUT): Once more, consider the fragment of the student grant example of Figure 5.2. As earlier seen in Example 3.3, operations op1 and op2 must not be grouped into an activity, because such an activity would have redundant input elements. This can also be identified based on the prerequisites that differ between op1 and op2. Table 5.2 shows that data element i41 is one of these differing prerequisites. Because i41 is a conditional data element, the alternative operations require the computation of different important data elements. Hence, the alternative operations must not be grouped into one activity, but rather into multiple activities.

Algorithm 4 provides a straightforward implementation that utilizes prerequisites to identify if the alternative operations of a data element result in an activity with redundant input. The implementation identifies all occurrences of redundant input. However, in certain cases, even though alternative operations require different data elements, this does not result in an activity with redundant input. Appendix B illustrates these exceptional cases, and describes an extended implementation that identifies them.

Algorithm 4 Identification of redundant input

```
1:  Input1: PDM
2:  Input2: goals          ▷ Data elements identified by Algorithms 1 and 2
3:  result = ∅
4:  for d ∈ goals do
5:      for altop1 ∈ predecessors(d) do
6:          for altop2 ∈ predecessors(d) do
7:              difference = prereqs(altop1) − prereqs(altop2)
8:              if {difference} ∩ {goals} ≠ ∅ then
9:                  result.add(d)
10: return result
```

When a set of operations would result in an activity with redundant input, the set must be grouped into multiple activities. For each alternative operation in the set, a new activity is created that contains this operation. Although it is not encountered in the cases available

in literature, other operations may remain in the set. It is possible to simply assign these operations to new activities in the same way as the alternative operations. The interested reader is referred to Appendix B for a more complex alternative to this assignment.

After the operations that lead to redundant input have been separated, the activity composition is completed. The result is a number of sets of operations that each represent an activity. For the student grants case, the result of the algorithm is the set of activities shown in Figure 2.6. If a PDM definition contains more information than the structural data-flow relations, the proposed activities can be further refined. This is described in Section 5.1.5.

5.1.5 *Refinement*

As introduced in Section 4.3, the activities that have been generated based on structural properties can be further refined when information about resource requirements and information sources is available. Refinement based on external information sources is performed by creating multiple leaf activities. Each leaf activity then contains the leaf operations that are related to the same information source.

The resource perspective can be incorporated in the activities by introducing *hand-over data elements*. Recall from Section 4.3.1 that these data elements are produced by an operation that requires one type of resource, while it is the input element of an operation that requires a different resource class. By extending Algorithm 1 such that it also identifies these hand-over elements, the resource perspective is incorporated in the implementation. Hence, extending the approach so that it includes these two types of refinement is fairly straightforward.

5.2 TOOL SUPPORT

The algorithm described in Section 5.1 and Appendix B has been implemented as a *plug-in* for the ProM6-framework[1]. ProM is an extensible framework that supports process mining, as well as PBWD. The plugin is available as the *PDMAggregation* package, through the built-in package manager of ProM6. A screenshot of this plug-in is provided in Figure 5.3. This section describes how the software supports modellers when composing activities for a given PDM.

5.2.1 *Input*

The input that is required for the plug-in, is an Extensible Markup Language (XML) representation of a PDM. The required XML schema definition (XSD) has been defined in [35], and is provided in Ap-

1 www.promtools.org/prom6

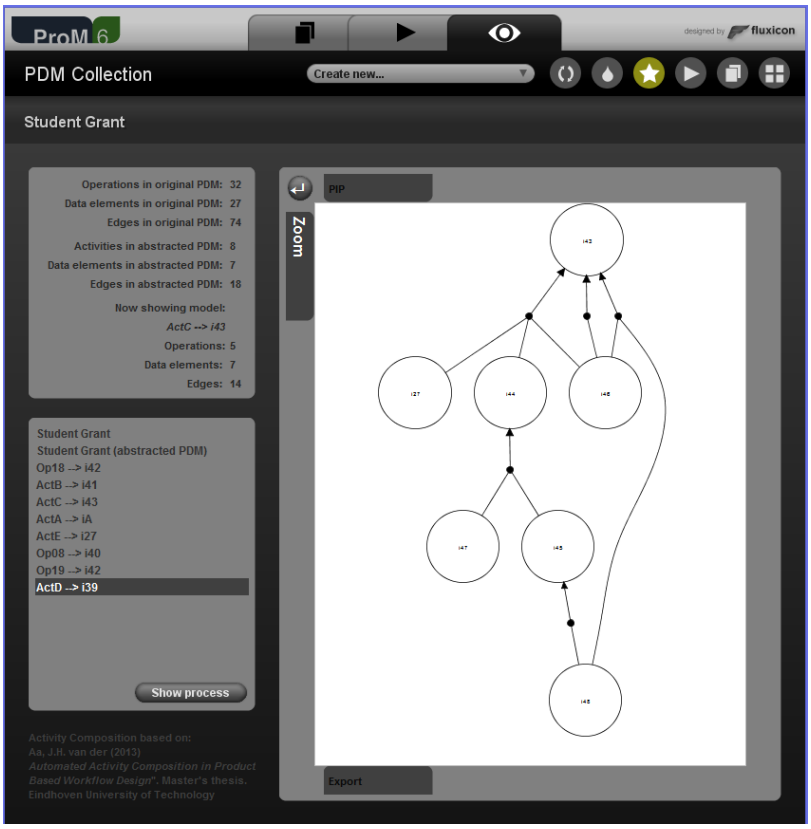


Figure 5.3: Screenshot of the ProM6-plugin

pendix C. A single adjustment is made to the original XSD: support for the inclusion of information sources is added as an optional attribute of operations. This addition allows refinement of leaf activities, as described in Section 4.3.2.

The minimal information that is required for the plug-in is a structural description of a PDM. An XML definition must therefore always contain a list of data elements, and a list of operations. Each operation is defined by specifying its input and output elements. A PDM can be further extended with optional information, such as, resource requirements, information sources, execution costs, and processing time.

5.2.2 *Customization*

The software allows modellers to fully automatically generate a set of activities that adhere to the guidelines of Chapter 4. The quantitative evaluation of Chapter 6 shows that these activities are good approximations of activities that have been manually composed by experienced modellers. However, since activity composition is a subjective task, the proposition does not imply that the generated activities represent the only correct activity design. Modellers can have differing preferences, that result in varying activity designs. For this reason, the

software includes functionality that supports customization. Through this function, users can incorporate their preferences in the activity generation. One can mark any set of data elements that are considered to be important. The tool then generates an activity design such that activities in the design produce the elements that are marked by the user. The operations that are assigned to each activity are the semantically related operations that are identified by Proposition 2. In order to ensure that the generated activities are valid, the implementation may introduce reference data elements when required.

The customization function thus allows users to incorporate their preferences in the generated activities. Furthermore, since the generation of activities takes mere seconds, the tool is ideal for iterative refinement. A modeller can generate activities until a design best meets his preferences.

5.2.3 *Results*

The implementation provides users with two types of results. First and foremost, a set of activities that adhere to the proposed design guidelines is computed. Secondly, the implementation produces an abstracted version of the original PDM. In this version, each activity replaces its set of underlying operations in the PDM. This abstracted PDM provides an overview of the most important data elements, their interrelations and possible executions paths. It is thus a very useful means to gain a quick overview of the most important information in a workflow. Figure 5.4 shows the abstracted PDM of the student grants case. It clearly shows different paths to compute the root element, and illustrates the four types of grant. The abstracted PDM furthermore forms the basis for the design of process models, either manually or by applying process generation algorithms (see e.g. [38]).

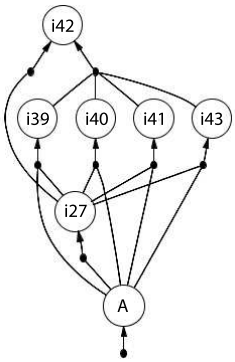


Figure 5.4: Abstracted PDM of the student grants case

The ProM plug-in is hence a means that allows users to quickly generate, view and customize activities for a given PDM. By enabling this tool support, this research presents the first available tool to support activity composition for workflow processes.

6 VALIDATION

The proposed activity composition guidelines call for validation. The goal of this validation is to determine how well the guidelines approximate designs made by modelling experts. This chapter presents an evaluation of the three propositions posed in Chapter 4. These propositions are quantitatively evaluated based on validation metrics from the field of information retrieval.

The chapter is structured as follows. Section 6.1 describes the setup used to evaluate the propositions. Section 6.2 presents the quantitative results of the validation. Finally, this chapter is concluded with a discussion of the results in Section 6.3.

6.1 SETTING

The validation is performed by comparing activity designs that are automatically generated based on the propositions, referred to as *proposed designs*, with manual activity designs derived from literature or experts, referred to as *solutions*. Five cases have been selected from literature: the student grant case of Sect. 2.2 and [36], the unemployment case of [24], the bicycle case of [6], the fireworks case of [35] and a United States student grant case. The cases range from a PDM with 32 operations and 27 data elements, to a PDM of 81 operations and 46 data elements. In total, there are eleven solutions available for the five cases. Recall from Chapter 4 that the student grant and unemployment cases have been used during the design of the composition guidelines. The validation results for these cases may thus be considered biased. It is therefore that, in order to present objective scores, the results for these cases are considered separately, as *training set* results. The three unbiased cases then form the *validation set*.

This section describes the five cases and the solutions that are derived from experts. For each solution, the granularity, as well as the perspectives that are emphasised by the modellers are denoted. Appendix D further presents the data that comprises these literature solutions.

6.1.1 *Student Grants (NL)*

The Dutch student grants case (SGNL) is derived from [35], and has been presented in Section 2.2 as the running example for this thesis. It considers the process that deals with requests for governmental student grants in the Netherlands. The PDM of the student grants

case contains 32 operations, 27 data elements and 74 arcs. Three available designs have been derived from literature, these are all derived from [35] and summarized in Table 6.1. The manual designs emphasise the aspect of process granularity through the use of *Cohesion & Coupling* metrics; these quantitative metrics are considered in Section 1.2. It is interesting to note that the automatically generated solution (see design 2 of Section 2.3) is *completely identical* to solution SGNL2. Because the SGNL case and its solutions have been studied during the design of the composition guidelines, this case is assigned to the training set of the validation.

ID	SOURCE	# OF ACTIVITIES	FOCUS
SGNL1	[35, p.200]	9	Optimization of granularity based on *Cohesion & Coupling* metrics.
SGNL2	[35, p.200]	8	Coarser-grained alternative design of SGNL1.
SGNL3	[35, p.200]	12	Finer-grained alternative design of SGNL1.
SGNL4	generated	8	Generated based on propositions.

Table 6.1: Activity designs for the SGNL case

6.1.2 *Unemployment Benefits*

The unemployment benefits case (UNEM) describes the process of awarding unemployment benefits in the Netherlands. It is considered in [24, 35]. The workflow process is a highly automated process, in which 37 of the 51 operations are executed automatically. The PDM of the process further consists of 42 data elements and 127 arcs. The available process designs are manual designs that emphasise efficient process execution. These are described in Table 6.2. It is interesting to note that the UNEM case is the only case for which it is explicitly stated that the leaf elements are retrieved from multiple information sources.

Because the UNEM case and its solutions have been studied during the design of the composition guidelines, this case is assigned to the training set of the validation.

ID	SOURCE	# OF ACTIVITIES	FOCUS
UNEM1	[24, p.216]	15	The design is driven by a minimization of cost, minimum number of contact with claimant, and usage of the case assignment heuristic.
UNEM2	[35, p.165]	13	This activity design is an adapted version of UNEM1 that further aims to optimize process granularity based on *Cohesion & Coupling* metrics.
UNEM3	generated	17	Generated based on propositions.

Table 6.2: Activity designs for the UNEM case

6.1.3 *Student Grants (US)*

The United States student grants case (SGUS) represents an adaptation of the SGNL case. The original case has been adapted in order to better resemble the student grant system in the U.S. Its informational structure is significantly more complex. It consists of 48 operations, 45 data elements and 166 arcs. The PDM and its respective activity designs are part of a United States Ministry of Defence course on business process modelling, specifically on Business Process Model and Notation (BPMN). This material is confidential and therefore unpublished.

There are three alternative designs available for this case. They have been designed by the trainer of the BPMN track, who is considered a subject matter expert. The three designs resemble process designs with varying granularities. These are summarized in Table 6.3.

6.1.4 *Bicycle Manufacturing*

The bicycle manufacturing case (BIC) describes the administrative part of a cross-departmental bicycle production process. The case is derived from [6]. Its information structure is very thin, and thereby differs greatly from the other four cases. The PDM consists of 36 operations, 29 data elements and 87 arcs.

The bicycle case is the only considered case for which there were originally no activity designs available. Therefore, two modellers were

ID	SOURCE	# OF ACTIVITIES	FOCUS
SGUS1	unpublished	5	A very coarse-grained process that is designed for maximum process maintainability, but which hides many details.
SGUS2	unpublished	11	A complex design with maximum detail.
SGUS3	unpublished	9	A design with intermediate granularity and structural complexity.
SGUS4	generated	12	Generated based on propositions.

Table 6.3: Activity designs for the SGUS case

asked to manually design activities. Both modellers are well-versed in PBWD and very familiar with the bicycle case.

The modellers noted that they designed activities that each addressed an identifiable part of the process. Furthermore, they utilized implicit resource information from the case description in [6]. Based on this information, each activity has been designed such that it can be performed by a single resource. The granularity of these designs is described in Table 6.4.

ID	SOURCE	# OF ACTIVITIES	FOCUS
BIC1	unpublished	15	The focus of this design was on semantics and resource requirements.
BIC2	unpublished	15	The focus on this design is similar to that of BIC1.
BIC3	generated	14	Generated based on propositions.

Table 6.4: Activity designs for the BIC case

6.1.5 *Fireworks*

The fireworks case (FW) presents an application of PBWD on a workflow process of a Dutch governmental institution. The case is derived from [39]. The process under consideration assesses requests for permission to ignite fireworks at special events. The PDM of the case consists of 81 operations, 46 data elements and 262 arcs. The ratio of operations to data elements reveals that there are many alternative operations in the PDM. There is a notably high number of alternative paths that lead to a value for the end product: 496.

Only one process design for this case has been found in literature. It is initially described in [39], but presented in more detail in [35]. The process design consists of activities that represent distinct subprocesses in the workflow. The characteristics of this design, as well as that of the generated design are described in Table 6.5.

ID	SOURCE	# OF ACTIVITIES	FOCUS
FW1	[35, p.215]	9	The activities in the process design are composed such that they represent clear sub-processes.
FW2	generated	9	Generated based on propositions.

Table 6.5: Activity designs for the FW case

6.2 RESULTS

This section describes a quantitative evaluation of the propositions against the literature solutions presented in Section 6.1. There are no resource specifications available for the five cases. Refinement through resources, such as described in Section 4.3.1, is therefore excluded. The UNEM case is the single case in which information sources are explicitly described. Refinement of the leaf activities based on these information sources is hence included in the validation of this case.

6.2.1 *Validation of Important Data Elements*

Proposition 1 proposes that important data elements in a PDM can be identified based on five structural patterns. This proposition is evaluated by comparing the identified elements with the output elements of activities in the solutions. Hence, the validation is based on the assumption that the solutions, similar to the proposed designs, are

also designed such that they produce a value for an important data element.

ASSUMPTION 1: The activities in literature solutions are designed such that the output element is an important data element.

Assumption 1 appears to be valid for all but the solutions of the unemployment case. As denoted in Section 6.1.2, these solutions are explicitly defined to emphasise execution efficiency and do not consider semantics. For this reason, the UNEM case is not considered in the validation of Proposition 1.

6.2.1.1 *Setup*

Proposition 1 is evaluated by comparing the output elements in the proposed designs with the output elements of activities in the solutions. The performance is quantified by computing *precision*, *recall* and *F-score*. These scores are, respectively, formalized in equations 6.1, 6.2, and 6.3. In these equations, a *true positive* (tp) denotes a data element that is correctly identified as important, whereas a *false positive* denotes a data element that is incorrectly identified as important. *True negative* (tn) and *false negative* (fn), respectively, represent correct and incorrect predictions for data elements that are identified as not important.

The data used for the evaluation of Proposition 1 is described in Appendix D.

$$\text{Precision} = \frac{tp}{tp + fp} \tag{6.1}$$

$$\text{Recall} = \frac{tp}{tp + tn} \tag{6.2}$$

$$\text{F-score } = 2 * \frac{\text{precision} * \text{recall}}{\text{precision} + \text{recall}} \tag{6.3}$$

Precision indicates the fraction of identified important data elements that match output elements in the solutions, whereas recall is the fraction of output elements in literature that were also identified by the proposed designs. Finally, the F-score is the harmonic mean of the precision and recall measures ([2]).

6.2.1.2 *Results*

The results[1] for Proposition 1 are presented in Table 6.6. The average scores on all three measures are high, with notable averages of

1 For cases with multiple solutions, the average score is presented.

	SGNL	UNEM	TRAIN. AVG.	SGUS	BIC	FW	VALID. AVG.
Precision	1.00	n/a	1.00	0.86	0.89	0.86	0.87
Recall	0.84	n/a	0.84	0.86	0.62	0.89	0.79
F-score	0.90	n/a	0.90	0.83	0.72	0.89	0.81

Table 6.6: Validation of Proposition 1

1.00 (training set) and 0.87 (validation set) for precision. This implies that the patterns are valuable indicators of important data elements for the solution set. For six out of the nine included solutions, the score on either precision or recall is 1.00. This suggests that the proposed designs balance between fine-grained solutions (indicated by high precision) and coarse-grained solutions (high recall). As precision dominates recall, the automatically generated designs are on average more coarse-grained than the solutions.

6.2.2 *Validation of Semantic Relatedness*

The second proposition in this paper states that operations are semantically related if they have the same associated data element and if the aggregation of those operations does not result in an activity with redundant input elements. To validate if such operations are indeed semantically related, proposed activity designs are compared with activities from the solutions. In order to asses semantic relatedness, it is assumed that the activities in the solutions consist of semantically related operations.

ASSUMPTION 2: The activities in the literature solutions consist of operations that are semantically related to each other.

As for the evaluation of Proposition 1, this assumption once again results in the exclusion of the unemployment solutions, since these solutions explicitly do not consider semantic relatedness.

6.2.2.1 *Setup*

Proposition 2 is evaluated by comparing the contents of activities in the proposed designs with those of activities in the solutions. In order to evaluate the performance of Proposition 2 independent of the performance of Proposition 1, the proposed designs are generated based on the output elements derived from the literature solutions. As each activity represents a cluster of operations, clustering performance measures are utilized to evaluate the proposition. The used indices are the *Rand index* and *Jaccard index*.

The Rand index computes how similar the activities, i.e. clusters, in the proposed designs are to the activities in the literature solu-

	SGNL	UNEM	TRAIN. AVG.	SGUS	BIC	FW	VALID. AVG.
Jaccard	0.96	n/a	0.96	0.94	0.74	0.95	0.88
Rand	0.99	n/a	0.99	0.98	0.97	0.95	0.97

Table 6.7: Validation of Proposition 2

tions [10]. The Rand index measures both correctly identified similarity, i.e. true positives, as well as correctly identified dissimilarity, i.e. true negatives. The index is shown in Equation 6.4.

$$\text{Rand index} = \frac{tp + tn}{tp + fp + tn + fn} \tag{6.4}$$

The Jaccard index [18] is similar to the Rand index, with the difference that the former only considers correctly identified similarity. The Jaccard index is included because it can, in some cases, reveal information that is not shown by the Rand index. Such situations can occur for fine grained activity designs, in which a high amount of true negatives may conceal a low number of true positives.

$$\text{Jaccard index} = \frac{tp}{tp + fp + fn} \tag{6.5}$$

The possible scores on both indices range from 0.0 to 1.0. A score of 1.0 means perfect similarity between the proposed design and a literature solution. A score of 0.0 occurs when, for example, all operations are combined into one activity in the proposed design, whereas the solution consists of separate activities for each operation.

6.2.2.2 *Results*

Table 6.7 provides the scores for the Jaccard and Rand indices. With average scores, respectively, close to and well-above 0.9, the Jaccard and Rand indices indicate that the automatically generated designs show a high similarity to the manual designs of modelling experts. The majority of the differences are caused by the ban on redundant input elements in the proposed designs. This constraint is not adhered to by several of the solutions. By excluding differences that result from the avoidance of redundant input elements, the average Rand-index is higher than 0.98. In other words, the proposed designs are then nearly identical to the solutions. This implies that consideration of associated data elements very well approximates activities designed by human modellers, while the largest differences occur due to differing definitions of correctness.

	SGNL	UNEM	TRAIN. AVG.	SGUS	BIC	FW	VALID. AVG.
Jaccard	0.89	0.46	0.68	0.88	0.48	0.95	0.77
Rand	0.98	0.87	0.93	0.96	0.91	0.95	0.94

Table 6.8: Validation of Proposition 3

6.2.3 *Validation of Well-Designed Activities*

Unlike the validation of the previous two propositions, the validation of Proposition 3 is performed without assumptions about the activities in the solutions.

6.2.3.1 *Setup*

The validation of Proposition 3 is performed in the same way as the validation of Proposition 2. The difference is, however, that the proposed designs are here generated based on the data elements identified by Proposition 1, rather than based on those derived from the literature solutions.

6.2.3.2 *Results*

As shown in Table 6.8, the scores for the Rand-index range from 0.81 to 0.98. They are thus relatively stable, when compared to the Jaccard index. Values for the latter index are spread, ranging from 0.46 to 0.95. A high Rand index and low Jaccard index on the same solution implies that the proposed design is more coarse-grained than the solution. This further reveals that the occurrence of *false negatives* in such designs, i.e. operations that are clustered in the solution but not in the proposed design, is very limited.

6.3 DISCUSSION

The validation of the three propositions has shown that the automatically generated designs show a high similarity to the manually designed solutions found in literature. The proposed guidelines thus approximate the design choices made by experienced modellers.

The evaluation of Proposition 1 reveals that the generated designs are on average more coarse-grained that the solutions. This suggests that even though the proposed patterns are valid identifiers of important data elements, experts also take other factors into account when determining the proper granularity of activities. A clear example is found in the bicycle case. Since this case lacks an explicit resource description, work hand-overs are not captured by the generated solutions. Manual modellers, by contrast, incorporate these hand-over

intuitively. This then results in a larger process granularity for the proposed design, due to the limited amount of explicit information that is available to generate the activities. The solutions of the Dutch student grant case also show a second reason for differing granularities. There, the solutions vary the number of activities that are related to the calculation of the supplementary grant. Intuitively, the modeller considers this sub-process to be too large for a single activity [35]. For this reason, SGNL1 and SGNL3 split this sub-process into multiple activities. Therefore, these two designs are considered to have a better granularity. This intuition is, however, not captured in the generated activities. Hence, the generated design is coarser-grained than SGNL1 and SGNL3.

Section 6.2.2 shows that the semantic relatedness of operations is well-approximated by Proposition 2. The only major differences between the proposed designs and the solutions occur due to differing notions of correctness. Section 3.3.2 justifies the avoidance of activities with redundant input because these constructs hide important information. By contrast, some solutions (SGNL1, SGNL3 and BIC1) do contain these constructs. The activities in these solutions are therefore not reproduced by the proposed guidelines.

The evaluation of Proposition 3 shows that its performance varies between the cases. The UNEM and BIC cases have relatively low scores on the Jaccard-index. For the bicycle case, this is due to differing granularity. As previously indicated, the solutions for this case, intuitively, included resource requirements. Since implicit resource information is not captured in the generated design, this design is coarser-grained. The generated designs show the least similarity to the solutions of the unemployment case. The solutions execute the automated operations as soon as they are enabled. On the contrary, the proposed designs only execute the automated operations when required. Since these automated operations encompass a majority of the operations, this difference has a significant impact on the quantitative results. However, the implications are arguably irrelevant, as the execution of automated operations is instantaneous and costless. When only non-automated operations are considered, the proposed design and the solutions of the unemployment case are *completely identical*.

The validation shows that the automatically generated designs are good approximations of designs by experienced modellers. The causes of the most important differences have furthermore been shown to be clearly identifiable. These differences occur either due to differing preferences, or because of the minimal amount of process information that is available for the generation of the designs.

7 CONCLUSION

The research in this thesis provides automated support for the composition of workflow activities. The proper composition of activities is important for the design of efficient and understandable workflow processes. However, without support, this task requires broad expertise and is time-consuming. The presented research provides such support through the introduction of fundamental guidelines for objective activity composition. It thereby enables non-experts to design proper activities. Because the guidelines are implemented in an automated approach, the time-consuming nature of activity composition is overcome. The implementation, furthermore, supports the iterative refinement of automatically generated activities. Through this functionality, modellers can incorporate their preferences in the generated activity designs.

To achieve the desired automated support, a broad range of factors that influence the quality of activity designs has been considered. This scoping phase resulted in the identification of *semantic relatedness* and *importance* of data-flow elements as the most promising factors to be incorporated in objective guidelines. It has been shown that these properties can be identified based on structural data-flow relations, captured in a PDM. The guidelines, furthermore, ensure efficient workflow execution. The activity designs avoid redundant, as well as deferred process execution. A quantitative evaluation has shown that the automatically generated activity designs are close approximates of activity designs manually created by experienced modellers. Since the majority of the cases used in this evaluation is based on existing business processes, the applicability of the guidelines in practice seems justified.

The close resemblance to manual activity designs is achieved despite a minimal amount of process data that is used to generate the activities. It has been shown that in cases were information beyond the structural data-flow relations is available, the generated activities can be further refined. The objective refinement of activities has been illustrated through the inclusion of additional guidelines that consider the resource perspective and external information sources.

The research presented in this thesis has been shown to successfully support the subjective and time-consuming task of manual activity composition with an objective and largely automated approach.

7.1 LIMITATIONS

Although the presented work shows promising results, it is subject to some limitations. Throughout this research, the most important limitation has been the number of available example cases. Few suitable PDMs and accompanying activity designs are found in literature. The available cases are based on existing business processes, and vary in complexity and domain. Even though this shows that the evaluation covers a diverse range of real processes, the strength of the evaluation can be increased by incorporating additional cases and activity designs. Furthermore, the available activity designs did not include explicit resource specifications. Therefore, the performance of the guidelines on workflows with a prominent resource dimension has not been evaluated. Aside from the impact on the validation strength, the limited number of cases may also have left structural patterns undiscovered. It is possible that other workflows reveal additional patterns that indicate the importance of data elements.

Other possibilities for improvement lay in the strictness of the design guidelines. The guidelines consider data element importance as a binary property; hence, all important data elements are regarded to be of equal significance. It is conceivable that an approach which recognizes different levels of importance, is more appropriate in certain cases. For example, for data-flow structures with a lot of important data elements, a more extensive classification could be used to create activities that encompass multiple, less important data elements. Strictness also plays a role in the avoidance of activities with redundant input elements. The prohibition of these activities is here justified because they hide important information and require the use of complex constructs. However, it has been found that experienced modellers sometimes deliberately ignore this notion in their activity designs. It is thus possible that there exist conditions for which modellers prefer designs that require those complex modelling constructs. If such conditions exist and are identified, the design guidelines may avoid redundant input less strictly, and thereby improve the quality of the generated activities.

7.2 FUTURE RESEARCH

This research has shown the feasibility of objective and automated support for the composition of workflow activities. This results in several opportunities for future research.

Firstly, the approach itself can be extended. Besides improvement of the existing guidelines, such as considered in the previous section, the set of guidelines can be extended by including more process information. Such additional guidelines may incorporate the factors that have been excluded here, due to a lack of objectivity or because they

depend on information beyond the definition of a PDM. If contextual information about a workflow is available, guidelines could, for example, apply a maximum processing time per activity, or avoid too complex activities. An extension furthermore presents opportunities to integrate principles of interaction design in the approach. These principles could be used to optimize activities for the workflow environment in which users execute them. Through such future research, the approach can be extended into one that encompasses all aspects of workflow activity design.

The presented research considers the design of activities for workflow processes. It does not assess the step that converts an activity design into a process model. Future work can focus on an approach which derives process models that are specifically fitted to the generated activities. For example, the activities are designed for high execution efficiency. Amongst others, they avoid redundancy and allow concurrent execution. A next step is then to design process models that emphasise such characteristics. The abstracted PDMs generated by the implementation provide an excellent starting point for automated process derivation. These abstracted versions already have a structure that is similar to that of process models. They furthermore capture all structural inter-dependencies that must be considered to create sound workflows. By extending the activity composition approach with a step that generates suitable process models, the transformation from data-flow structure to workflow process can be fully supported.

On a higher level, future work can consider the applicability of the composition guidelines in other parts of business process management. The area of *process mining* appears to be particularly interesting for this. It is the goal of process mining to extract information about processes from transaction logs. These logs contain large quantities of data about atomic processing steps, i.e. *events*. The sheer size of this data often distorts understandability of the results. The concepts proposed in this thesis may be able to identify and group related events. Thereby it could provide a higher level overview of information that is otherwise too large or complex to be comprehended. This makes the applicability of the guidelines in process mining a very promising direction for future research.

A

CORRECTNESS OF LEMMA 1

Lemma 1 proposes that each operation in a PDM is associated with exactly one important data element. This appendix provides a proof of this lemma.

It is here shown that each operation in a PDM has exactly one associated data element. An important data element is identified by any of the five structural patterns in Section 4.1. The most important structural pattern in the proof of this lemma is the pattern of *reference data elements*. Informally, Pattern 5 defines a reference data element as a data element that is an input element to multiple operations that are, directly or indirectly, involved in the computation of different sets of important data elements. By including the notion of associated data elements, this definition is formalized in Definition A.1.

DEFINITION A.1 (REFERENCE DATA ELEMENT): A reference data element, is any data element d, for which d is an input element to at least two operations, op1 and op2, with g1 as the associated element of op1, and g2 as the associated element of op2, such that $g1 \neq g2$.

The associated data element of an operation is formalized in Definition A.2.

DEFINITION A.2 (ASSOCIATED DATA ELEMENT): Given D : the set of important data elements. $d \in D$ is an associated data element of operation op, if there exists a path p from op to (but excluding) d, such that $p \cap D == \emptyset$.

Lemma 1 proposes that by this definition, each operation has exactly one associated data element.

PROOF OF LEMMA 1

An operation in a PDM always has at least one associated data element, because a PDM is *connected*, therefore for each operation op, there is a path from op to root, and *root* is an important data element, as defined in Pattern 1.

The property that each operation is associated with at most one data element, is determined as a *proof by contradiction*. This proof follows from the way reference data elements are identified in a PDM, given by Definition A.1. It is shown that, due to reference data elements, an operation cannot be associated with two (or more) data elements, and therefore must thus have at most one associated data element.

Assume that operation op has two distinct associated data elements, g1 and g2. Then by Definition A.2, there must exist path p1

from op to g1 and path p2 from op to g2, such that $p1 \wedge G = \emptyset$ and $p2 \wedge G = \emptyset$.

Operation op has exactly one output data element, therefore paths p1 and p2 start from the same data element and overlap for an arbitrary length.

Since $g1 \neq g2$, there must exist two operations, op1 and op2 that respectively produce g1 and g2. Therefore, there must exist a data element d that is used as an input elements by (at least) two operations, op3 and op, such that op3 is associated with g1 and op4 is associated with g2. However, by Definition A.1, data element d is a reference data element. Therefore, op cannot be associated with g1 and g2, since $p1 \wedge g1 \neq \emptyset$ and $p2 \wedge g2 \neq \emptyset$.

Since the aforementioned shows that an operation always has at least one, and not more than one associated data elements, Lemma 1 has been proven.

B

IMPLEMENTATION EXTENSIONS

Appendix B presents two atypical cases that can be encountered during activity composition. For both cases, their complexity is here considered to outweigh their impact on the implementation presented in Chapter 5. Therefore, the implementation abstracts from these cases in order to maintain understandability.

B.1 REDUNDANT INPUT EXCEPTION

As stated in Section 5.1.4, it is possible that multiple alternative operations, that require differing sets of important data elements, do not result in an activity with redundant input. This is possible, because Constraint 1 states that redundant input occurs when "it would be possible to execute the activity without first computing all of its input elements." Therefore, differing prerequisites may not always lead to redundant input. Example B.1 provides an example of this.

EXAMPLE B.1 (FALSE REDUNDANT INPUT): Consider Figure B.1, it depicts activity C1 of Section 2.3.2. This activity contains two alternative operations: op1 and op2. The former operation, op1, only requires element i37, whereas op2 requires a larger set of input elements. This set includes the reference data element i27. Therefore, the two alternative operations have differing sets of important data elements as their prerequisites. Hence, Algorithm 4 would identify this activity to have redundant input elements. However, if the entire student grant case of Figure 2.2 is inspected, it is clear that i27 must always be computed, irrespective of the alternative selected for activity C1. Therefore, the activity in Figure B.1, cannot result in the redundant computation of i27.

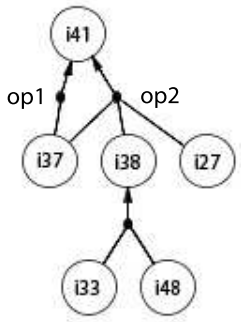


Figure B.1: Activity C1

To formalize the concept shown in Example B.1, *alternatives* are defined in Definition B.1. Example B.2 illustrates this notion.

DEFINITION B.1 (ALTERNATIVE): An alternative is any minimal set of operations and their output elements, such that the operations can be executed to create the root data element of a PDM.

EXAMPLE B.2 (ALTERNATIVES): Consider the PDM shown in Figure B.2. There are three possible alternatives for this PDM. These alternatives are described in Figure B.1. The alternatives represent those sets of operations that can be executed to compute the root element A. Any other set of operations is either unable to compute the root element, or is an extended version of one of the three alternatives. In the latter case, the set is not considered to be *minimal*.

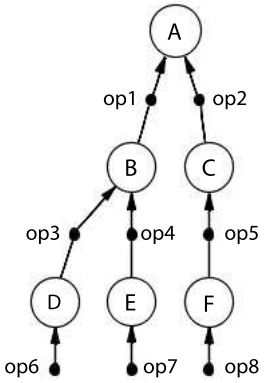


Figure B.2: Alternatives

ID	CONTENT
1	{op1, op3, op6, A, B, D}
2	{op1, op4, op7, A, B, E}
3	{op2, op4, op8, A, C, F}

Table B.1: Elements per alternative

An activity has redundant input if it can be executed without computing all its input elements. This can be identified by utilizing the concept of alternatives. Given the alternative operations that produce an element d, and a differing prerequisite data element d2. An activity that produces d then has redundant input if and only if d2 does *not* occur in all the alternatives that contain d. For example, in Example B.1, there is no alternative in the student grant case that contains conditional data element i41 but not element i27. Therefore, computation of i27 is never redundant for the activity of Figure B.1. Algorithm 5 implements this notion; it resembles an extension of Algorithm 4. The algorithm uses *alternatives(x)* to denote the alternatives that contain element x.

B.2 ASSIGNING REMAINING OPERATIONS

When a set of operations results in an activity with redundant input, the operations must be assigned to multiple activities. Section 5.1.4 describes the most straightforward approach for this assignment: create a new activity for each operation in the set. This approach ensures that the resultant activities do not have redundant input and further-

Algorithm 5 Extended identification of redundant input

```
1: Input1: PDM
2: Input2: goals          ▷ Data elements identified by Algorithms 1 and 2
3: result = ∅
4: for d ∈ goals do
5:     for altop1 ∈ predecessors(d) do
6:         for altop2 ∈ predecessors(d) do
7:             difference = prereqs(altop1) − prereqs(altop2)
8:             differingGoals = difference ∩ goals
9:             for d2 ∈ differingGoals do
10:                if (alternatives(d) − alternatives(d2)) ≠ ∅ then
11:                    result.add(d)
12: return result
```

more is generally well-applicable in practice. However, it is not be the best approach for some cases. If the set of operations contains more than just a number of alternative operations, other approaches are more suitable. This section describes such an approach; it attempts to split the set of operations into activities that best resemble the notion of semantic relatedness, as proposed in Proposition 2.

First, for each alternative operation in the set, a new activity is created that contains this operation. The separation of the alternative operations ensures that these activities do not have redundant input elements. Any operations that remain, i.e. those that are not alternative operations, are then considered. There are two cases to assign these *remaining operations*. Firstly, if a remaining operation (op1) is a prerequisite of only one of the alternative operations, then op1 is assigned to the activity that contains this alternative operation. Otherwise, op1 is assigned to an activity with other operations that are required by the same alternative operations, if any. Example B.3 illustrates both cases.

EXAMPLE B.3 (SEPARATING ALTERNATIVE OPERATIONS): Assume that the operations in Figure B.3 form an activity that results in redundant input. The operations must therefore be assigned to multiple activities. To accomplish this, firstly, two activities are created for the alternative operations, i.e. one that contains op1, the other with op2. Then, $\{op3, op4, op5\}$ are the remaining operations. op3 is a prerequisite of only one alternative operation: op1. Hence, op3 is assigned to the same activity as op1. op4 and op5 are, however, both prerequisites of alternative operations op1 and op2. Therefore, a third activity is created that consists of these operations. The resultant activities are thus: $\{op1, op3\}$, $\{op2\}$ and $\{op4, op5\}$.

Through the assignment illustrated in the example, the operations are grouped into activities that best resemble the notion of semantic

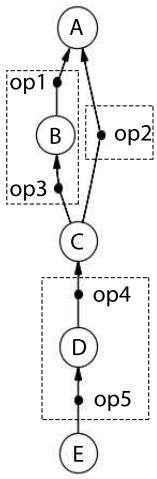


Figure B.3: Separating alternative operations

relatedness that has been earlier proposed. Algorithm 4 provides an implementation of this assignment approach.

Algorithm 6 Separating alternative operations

```
1:  Input1: altOps
2:  Input2: remainingOps
3:  activities = map<id, set>                    ▷ map from id to operations
4:  for altOp ∈ altOps do
5:      activities(altOp) = {altOp}              ▷ create new activity
6:  for op1 ∈ remainingOps do
7:      op1Alts = involvedWith(op1) ∩ altOps
8:      if size(op1Alts) == 1 then
9:          altOp = op1Alts.pop()                ▷ retrieve the alternative operation
10:         activities(altOp).add(op1)
11:     else
12:         activities(op1) = {op1}
13:         for op2 : remainingOps do
14:             op2Alts = involvedWith(op2) ∩ altOps
15:             if op1Alts == op2Alts then
16:                 activity.add(op2)
17:                 remainingOps.remove(op2)
18: return activities
```

C

XML DEFINITION OF A PDM

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="
    unqualified">
<xs:element name="PDM">
<xs:complexType>
    <xs:sequence>
        <xs:element ref="Name"/>
        <xs:element ref="DataElement" maxOccurs="unbounded"/>
        <xs:element ref="Resource" maxOccurs="unbounded"/>
        <xs:element ref="Source" maxOccurs="unbounded"/>
        <xs:element ref="Operation" maxOccurs="unbounded"/>
        <xs:element ref="RootElementRef"/>
    </xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="DataElement">
    <xs:complexType>
        <xs:sequence minOccurs="0">
            <xs:element ref="Data" minOccurs="0"/>
        </xs:sequence>
    <xs:attribute name="DataElementID" type="xs:ID" use="
        required"/>
    <xs:attribute name="Description" type="xs:string" use="
        optional"/>
    </xs:complexType>
</xs:element>
<xs:element name="Resource">
    <xs:complexType>
        <xs:attribute name="ResourceID" type="xs:ID" use="
            required"/>
        <xs:attribute name="Description" type="xs:string" use=
            "optional"/>
    </xs:complexType>
</xs:element>
<xs:element name="Source">
    <xs:complexType>
        <xs:attribute name="SourceID" type="xs:ID" use="
            required"/>
        <xs:attribute name="Description" type="xs:string" use=
            "optional"/>
    </xs:complexType>
</xs:element>
<xs:element name="Operation">
    <xs:complexType>
        <xs:sequence>
                <xs:element name="Input">
                    <xs:complexType>
                        <xs:sequence>
```

```
                        <xs:element name="DataElementRef"
                            type="xs:IDREF"
                        minOccurs="0" maxOccurs="unbounded
                            "/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
            <xs:element name="Output">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="DataElementRef" type
                            ="xs:IDREF"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
            <xs:element name="Cost" minOccurs="0"/>
            <xs:element name="Time" minOccurs="0"/>
            <xs:element name="Condition" minOccurs="0"
                maxOccurs="unbounded"/>
            <xs:element name="Probability" minOccurs="0"/>
            <xs:element name="ResourceRef" type="xs:IDREF"
                minOccurs="0"/>
            <xs:element name="SourceRef" type="xs:IDREF"
                minOccurs="0"/>
            <xs:element ref="Data" minOccurs="0"/>
        </xs:sequence>
        <xs:attribute name="OperationID" type="xs:ID" use="
            required"/>
        <xs:attribute name="Description" type="xs:string" use=
            "optional"/>
    </xs:complexType>
</xs:element>
<xs:element name="Data"/>
<xs:element name="RootElementRef" type="xs:IDREF"/>
<xs:element name="Name"/>
</xs:schema>
```

D

VALIDATION DATA

Appendix D presents the data that is used in the validation of Chapter 6. Table D.1 contains the important data elements that are derived from the literature solutions. The values in this table are used for the quantitative evaluation of Proposition 1. The remaining five tables in this appendix describe the activity designs found in literature. These values are used during the evaluation of Proposition 2 and Proposition 3.

CASE	DESIGN	OUTPUT DATA	PRECISION	RECALL	F-SCORE
SGNL	SGNL0[a]	i27, i39, i40, i41, i42, i43, leaf	n/a	n/a	n/a
	SGNL1	i27, i30, i39, i40, i41, i42, i43, leaf	1	0,88	0,93
	SGNL2	i27, i39, i40, i41, i42, i43, leaf	1	1	1
	SGNL3	i25, i27, i28, i29, i30, i39, i40, i41, i42, i43, leaf	1	0,64	0,78
SGUS	SGUS0	i26, i28, i36, i39, i40, i45, leaf	n/a	n/a	n/a
	SGUS1	i26, i40, i45, leaf	0,57	1	0,72
	SGUS2	i26, i27, i28, i32, i36, i39, i40, i42, i45	1	0,7	0,82
	SGUS3	i26, i27, i28, i39, i40, i42, i45, leaf	1	0,88	0,93
BIC	BIC0	i2, i11, i12, i15, i22, i24, i25, i27, i28, i29, leaf	n/a	n/a	n/a
	BIC1[b]	i2, i7, i8, i9, i10, i11, i12, i15, i22, i23, i24, i25, i26, i27, i28, i29	0,82	0,56	0,67
	BIC2	i2, i7, i9, i10, i11, i12, i15, i22, i23, i24, i25, i26, i27, i28, i29	0,91	0,67	0,77
FW	FW0	i1, i2, i12, i13, i14, i15, i16, i40, leaf	n/a	n/a	n/a
	FW1	i1, i2, i10, i12, i13, i14, i15, i16, leaf	0,89	0,89	0,89

Table D.1: Data used in the validation of Proposition 1

a *leaf* denotes the output element of a leaf activity, is such an activity is present in the given activity design.

b One of the activities in this design has three output elements. All three are here counted as important data elements.

DESIGN	ACTIVITIES
SGNL1	A = {op27, op28, op29, op30, op31, op32, op33, op34, op35, op36, op38}, B = {op20, op21, op22, op23, op25}, C = {op08}, D = {op01, op02, op03, op05}, E = {op06, op07}, F = {op15, op17, op24}, G = {op18}, H = {op19}, I = {op10, op11, op12, op13};
SGNL2	A = {op19} , B = {op01, op02, op03, op05, op10, op11, op12, op13, } , C = {op18} , D = {op07, op06} , E = {op23, op22, op21, op20, op25} , F = {op08} , G = {op17, op15, op24} , H = {op34, op38, op29, op28, op30, op31, op32, op33, op27, op35, op36};
SGNL3	A = {op27, op28, op29, op30, op31, op32, op33, op34, op35, op36, op38} , B = {op20, op21, op22, op23, op25} , C = {op08} , D = {op01} , E = {op06, op07} , F = {op15, op17, op24} , G = {op18} , H = {op19} , I = {op10, op11, op12, op13} , J = {op02} , K = {op03} , L = {op05};

Table D.2: Solutions SGNL case

DESIGN	ACTIVITIES
UNEM1[ab]	A = {O36, O38, O39, O41, O42, O46, O47, O48, O50, O51}, B = {O43, O35, O34, O37, O49, O45, O40, O44, Op01, Op02, Op03, Op04, Op05, Op06, Op07, Op08, Op09, Op25} , C = {Op10, Op30, Op32} , D = {Op17} , E = {Op18} , F = {Op19} , G = {Op20} , H = {Op16, Op26, Op27, Op29, Op33} , I = {Op14, Op15} , J = {Op11, Op24, Op28, Op31}, K = {Op13} , L = {Op21} , M = {Op23} , N = {Op12} , O = {Op22};
UNEM2[c]	A = {O36, O38, O39, O41, O42, O46, O47, O48, O50, O51}, B = {O43, O35, O34, O37, O49, O45, O40, O44, Op01, Op02, Op03, Op04, Op05, Op06, Op07, Op08, Op09, Op25} , C = {Op10, Op30, Op32} , D = {Op17} , E = {Op18} , F = {Op19} , G = {Op20} , H = {Op21} , I = {Op26, Op27, Op28, Op29, Op31, Op33} , J = {Op11} , K = {Op15} , L = {Op22} , M = {Op12} , N = {Op23} , O = {Op24} , P = {Op13, Op14, Op16};

Table D.3: Solutions UNEM case

a The original design contains two activities with operations {Op11, Op24, Op28, Op31}, these operations were assigned to a separate activity (J) in order to enable the calculation of the most objective scores.

b Operations Op12 and Op22 are originally not part of this design because these are considered to be inefficient. However, in order to properly calculate validation scores, these are included in activities N and O.

c The original designs does not include operations Op13, Op14 and Op16. Since these are connected, an additional activity (P) that consists of these operations is added for completeness.

DESIGN	ACTIVITIES
SGUS1	A = {O17, O11, O16, O08, O15, O23, O14, O19, O18, O12, O04, O05, O06, O07, O13, O01, O02, O03, O10, O22, O09, O20, O21} , B = {OpID02, OpID03} , C = {OpID04, OpID05, OpID06, OpID07, OpID08, OpID09, OpID10, OpID11, OpID12, OpID13, OpID14, OpID15, OpID16, OpID17, OpID18, OpID19, OpID20} , D = {OpID21, OpID22, OpID23, OpID24, OpID25} , E = {OpID01};
SGUS2	A = {O17, O11, O16, O08, O15, O23, O14, O19, O18, O12, O04, O05, O06, O07, O13, O01, O02, O03, O10, O22, O09, O20, O21} , B = {OpID01} , C = {OpID02, OpID03} , D = {OpID04} , E = {OpID05, OpID11} , F = {OpID07} , G = {OpID06, OpID08, OpID09,OpID10, OpID12,OpID13, OpID14, OpID15, OpID16} , H = {OpID17, OpID18, OpID19, OpID20} , I = {OpID22, OpID23, OpID21, OpID24, OpID25};
SGUS3	A = {O17, O11, O16, O08, O15, O23, O14, O19, O18, O12, O04, O05, O06, O07, O13, O01, O02, O03, O10, O22, O09, O20, O21} , B = {OpID02, OpID03} , C = {OpID04} , D = {OpID05, OpID11} , E = {OpID07} , F = {OpID06, OpID08, OpID09,OpID10} , G = {OpID12,OpID13, OpID14, OpID15, OpID16} , H = {OpID17, OpID18, OpID19, OpID20} , I = {OpID22, OpID23} , J = {OpID21, OpID24, OpID25} , K = {OpID01};

Table D.4: Solutions SGUS case

DESIGN	ACTIVITIES
BIC1	A = {op1, op2} , B = {op3, op4, op5, op6, op7} , C = {op8, op9} , D = {op10} , E = {op11, op12, op13, op14} , F = {op15} , G = {op16, op17, op18} , H = {op19, op20, op21, op22, op23, op24, op25} , I = {op26} , J = {op27, op28} , K = {op29} , L = {op30} , M = {op31} , N = {op32, op33} , O = {op34, op35, op36};
BIC2	A = {op1, op2, op6, op8, op10, op29} , B = {op3, op4, op5} , C = {op7} , D = {op9} , E = {op11, op12, op13, op14} , F = {op35} , G = {op15} , H = {op36} , I = {op16, op17, op18} , J = {op19, op20, op21, op22, op23, op24, op25} , K = {op26} , L = {op27, op28} , M = {op30} , N = {op31, op32, op33} , O = {op34};

Table D.5: Solutions BIC case

DESIGN	ACTIVITIES
FW1	A = {Op11} , B = {Op04, Op05, Op06, Op07} , C = {Op12, Op13} , D = {Op61, Op73, Op51, Op74, Op66, Op65, Op64, Op70, Op57, Op72, Op55, Op63, Op58, Op59, Op69, Op68, Op67, Op53, Op75, Op76, Op50, Op52, Op60, Op77, Op62, Op56, Op49, Op71, Op14, Op15, Op48} , E = {Op17, Op42, Op23, Op40, Op21, Op18, Op19, Op38, Op39, Op29, Op28, Op79, Op31, Op30, Op22, Op32, Op20, Op27, Op35, Op25, Op37, Op44, Op33, Op43, Op34, Op41, Op26, Op47, Op82, Op45, Op36, Op78, Op81, Op80, Op83, Op24, Op46} , F = {Op09, Op10} , G = {Op01, Op02} , H = {Op08} , I = {Op16};

Table D.6: Solutions FW case

BIBLIOGRAPHY

[1] W.M.P. van der Aalst. Re-engineering knock-out processes. *Decision Support Systems*, 30(4):451–468, 2001.

[2] Ricardo Baeza-Yates, Berthier Ribeiro-Neto, et al. *Modern information retrieval*, volume 463. ACM press New York., 1999.

[3] J.A. Buzacott. Commonalities in reengineered business processes: models and issues. *Management Science*, 42(5):768–782, 1996.

[4] D.J. Campbell. Task complexity: A review and analysis. *Academy of management review*, 13(1):40–52, 1988.

[5] P.J.N. de Crom and H.A. Reijers. Using prototyping in a product-driven design of business processes. In *Proceedings of the Open Enterprise Solutions: Systems, Experiences, and Organizations Conference*, pages 41–47, 2001.

[6] H.D. Diaz Garcia. Evaluation of data-centric process modeling approaches. Master's thesis, Eindhoven University of Technology, 2011.

[7] J.R. Hackman and G.R. Oldham. Motivation through the design of work: Test of a theory. *Organizational behavior and human performance*, 16(2):250–279, 1976.

[8] M. Hammer and J. Champy. *Reengineering the corporation: A manifesto for business revolution*. HarperBusiness, 2003.

[9] A.H.M. ter Hofstede, W.M.P. van der Aalst, M. Adams, and N. Russell. *Modern Business Process Automation: YAWL and its support environment*. Springer, 2009.

[10] Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of classification*, 2(1):193–218, 1985.

[11] C. Jarrett and G. Gaffney. *Forms that work: designing web forms for usability*. Morgan Kaufmann Pub, 2008.

[12] Frank Thomson Leighton. *Introduction to parallel algorithms and architectures*. Morgan Kaufmann San Mateo, Calif., 1992.

[13] H. Leopold, S. Smirnov, and J. Mendling. Refactoring of process model activity labels. *Natural Language Processing and Information Systems*, pages 268–276, 2010.

[14] Henrik Leopold, Jan Mendling, and Hajo Reijers. On the automatic labeling of process models. In *Advanced Information Systems Engineering*, pages 512–520. Springer, 2011.

[15] Maya Lincoln, Mati Golani, and Avigdor Gal. Machine-assisted design of business process models using descriptor space analysis. *Business Process Management*, pages 128–144, 2010.

[16] J. Mendling, H.A. Reijers, and W.M.P. van der Aalst. Seven process modeling guidelines (7pmg). *Information and Software Technology*, 52(2):127–136, 2010.

[17] Jan Mendling, Hajo A Reijers, and Jorge Cardoso. What makes process models understandable? In *Business Process Management*, pages 48–63. Springer, 2007.

[18] Glenn W Milligan. A monte carlo study of thirty internal criterion measures for cluster analysis. *Psychometrika*, 46(2):187–199, 1981.

[19] César Augusto L Oliveira, Ricardo Massa F Lima, Hajo A Reijers, and Joel Tiago S Ribeiro. Quantitative analysis of resource-constrained business processes. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 42(3):669–684, 2012.

[20] M.A. Ould. *Business Processes: Modelling and analysis for re-engineering and improvement*. Wiley London, 1995.

[21] A. Polyvyanyy, S. Smirnov, and M. Weske. Process model abstraction: A slider approach. In *Enterprise Distributed Object Computing Conference, 2008. EDOC'08. 12th International IEEE*, pages 325–331. IEEE, 2008.

[22] Artem Polyvyanyy, Sergey Smirnov, and Mathias Weske. Business process model abstraction. *Handbook on Business Process Management 1*, pages 149–166, 2010.

[23] H.A. Reijers. A cohesion metric for the definition of activities in a workflow process. In *Proceedings of the Eight CAiSE/IFIP8.1 International Workshop on Evaluation of Modeling Methods in Systems Analysis and Design*, pages 116–125, Velden, 2003.

[24] Hajo A. Reijers. *Design and control of workflow processes: business process management for the service industry*. Springer-Verlag, 2003.

[25] Hajo A. Reijers and S. Liman Mansar. Best practices in business process redesign: an overview and qualitative evaluation of successful redesign heuristics. *Omega*, 33(4):283–306, 2005.

[26] Hajo A. Reijers and I. Vanderfeesten. Cohesion and coupling metrics for workflow process design. *Business Process Management*, pages 290–305, 2004.

[27] Hajo Alexander Reijers, S. Limam, and Wil M. P. Van der Aalst. Product-based workflow design. *Journal of Management Information Systems*, 20(1):229–262, 2003.

[28] Shazia Sadiq, Maria Orlowska, Wasim Sadiq, and Cameron Foulger. Data flow and validation in workflow modelling. In *Proceedings of the 15th Australasian database conference-Volume 27*, pages 207–214. Australian Computer Society, Inc., 2004.

[29] A. Seidmann and A. Sundararajan. The effects of task and information asymmetry on business process redesign. *International Journal of Production Economics*, 50(2):117–128, 1997.

[30] S. Smirnov, H.A. Reijers, T. Nugteren, and M. Weske. *Business process model abstraction: theory and practice*. Number 35. Universitätsverlag Potsdam, 2010.

[31] S. Smirnov, H.A. Reijers, and M. Weske. A semantic approach for business process model abstraction. In *Advanced Information Systems Engineering*, pages 497–511. Springer, 2011.

[32] Workflow Management Coalition Specification. *Workflow Management Coalition, Terminology & Glossary (Document No. WFMC-TC-1011)*. Workflow Management Coalition Specification, February 1999. URL `http://www.wfmc.org/Download-document/WFMC-TC-1011-Ver-3-Terminology-and-Glossary-English.html`.

[33] Sherry X Sun, J Leon Zhao, Jay F Nunamaker, and Olivia R Liu Sheng. Formulating the data-flow perspective for business process management. *Information Systems Research*, 17(4):374–391, 2006.

[34] Nikola Trčka, Wil M. P. Van der Aalst, and Natalia Sidorova. Data-flow anti-patterns: Discovering data-flow errors in workflows. In *Advanced Information Systems Engineering*, pages 425–439. Springer, 2009.

[35] I. Vanderfeesten. *Product-Based Design and Support of Workflow Processes*. PhD thesis, Eindhoven University of Technology, 2009.

[36] I. Vanderfeesten, Hajo A. Reijers, and Wil M. P. Van der Aalst. Evaluating workflow process designs using cohesion and coupling metrics. *Comput. Ind.*, 59(5):420 – 437, 2008. ISSN 0166-3615. doi: 10.1016/j.compind.2007.12.007. URL `http://www.sciencedirect.com/science/article/pii/S0166361507001807`.

[37] Irene Vanderfeesten, Hajo A Reijers, and Wil M. P. Aalst. Case handling systems as product based workflow design support. *Enterprise Information Systems*, pages 187–198, 2009.

[38] Irene Vanderfeesten, Hajo A. Reijers, Wil M. P. Van der Aalst, and Jan Vogelaar. Automatic support for product based workflow design: generation of process models from a product data model. In *OTM 2010 Workshops*, pages 665–674. Springer, 2010.

[39] J. Vogelaar. The link between product data model and process model. Master's thesis, Eindhoven University of Technology, 2009.

[40] R.E. Wood. Task complexity: Definition of the construct. *Organizational behavior and human decision processes*, 37(1):60–82, 1986.

[41] M. Zapf, U. Lindheimer, and A. Heinzl. The myth of accelerating business processes through parallel job designs. *Information Systems and E-Business Management*, 5(2):117–137, 2007.