Computers in Industry 75 (2016) 162–177

Contents lists available at ScienceDirect

Computers in Industry

journal homepage: www.elsevier.com/locate/compind

Designing like a Pro: The automated composition of workflow activities

Han van der Aa [a,*], Hajo A. Reijers [a,b], Irene Vanderfeesten [c]

[a] Department of Computer Sciences, Vrije Universiteit Amsterdam, Faculty of Sciences, De Boelelaan 1081, 1081 HV Amsterdam, The Netherlands
[b] Department of Mathematics and Computer Science, Eindhoven University of Technology, PO Box 513, 5600 MB Eindhoven, The Netherlands
[c] School of Industrial Engineering, Eindhoven University of Technology, PO Box 513, 5600 MB Eindhoven, The Netherlands

ARTICLE INFO

Article history:
Received 4 July 2014
Received in revised form 17 April 2015
Accepted 28 April 2015
Available online 25 June 2015

Keywords:
Business process management
Workflows
Product Based Workflow Design
Activity composition
Work design

ABSTRACT

To design a workflow process that is efficient, meaningful, and understandable it is important to properly consider how to compose the activities it will consist of. In this paper, guidelines are presented for this exact purpose. These focus on the elementary data-processing steps that are at the core of a workflow process. The guidelines help to determine the relative importance of these data-processing steps as well as their relatedness, such that activities can be composed in a fully automated manner. We implemented this approach in freely available software. A thorough evaluation that incorporates real-life workflow designs indicates that the use of these guidelines leads to activities that closely resemble those designed by experienced modelers. As such, the proposed guidelines provide a proper and automated alternative to what is otherwise a complex and time-consuming task.

1. Introduction

Data-centric business processes pursue the production of an informational product. These processes, which are here referred to as *workflows*, are often found in the service domain. Typical informational products one can think of are, e.g. a mortgage contract, a decision on an insurance claim, or a personalized commercial offer. The structure of a workflow is to a large extent determined by the data-flow underlying the process. The data-flow comprises numerous elementary data processing steps, which all contribute to the ultimate computation of a desired end result. When designing a workflow, these elementary data processing steps are grouped into *activities*. Activities should be constructed in such a way that each of these represents a *logical piece of work* within a process [1]. An activity may or may not comprise multiple elementary processing steps. Creating a large activity, which is composed of many data processing steps, may often be preferable over including the elementary processing steps as separate activities. 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. Individually, each of these three steps may appear insignificant and overly fine-grained, while an activity that combines all three denotes an important and recognizable part of a workflow. In this paper, our attention is with the grouping of elementary data processing steps into activities. We will refer to this act as *activity composition*.

Properly carrying out activity composition is important, because the size and contents of activities affect three aspects of a process design: execution efficiency, experienced meaningfulness, and model understandability. First, activities can influence the *execution efficiency* of a process. Namely, 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 reduced flexibility caused by too many large activities [2,3]. Furthermore, incorrectly composed activities can result in the redundant or unnecessarily delayed execution of tasks. Second, activity composition affects the *experienced meaningfulness* of activities for those employees that execute these [4]. This aspect is associated with job satisfaction, motivation, and work performance [5,6]. As a result, more meaningful tasks can result in increased productivity. Finally, the design of activities also affects the *understandability* of process models. By grouping elementary steps into larger activities, the number of process model elements is reduced, which results in an increased understandability of the process [7]. Furthermore, activity designs that emphasize the important process steps, as well as properly depict the general process flow, can provide useful

* Corresponding author. Tel.: +31 20 59 87788.
E-mail address: j.h.vander.aa@vu.nl (H. van der Aa).

http://dx.doi.org/10.1016/j.compind.2015.04.005
0166-3615/

insights into the most important aspects of a process. Constructing such a quick process overview has been found to be a highly demanded use case for Business Process Model Abstraction [8].

Due to the impact that activity design can have on the quality of a workflow design, proper activity composition is a highly desirable task. Despite this importance, there exists a lack of support for this task [9]. Earlier work proposed metrics that objectively evaluate the quality of activity designs on the basis of job design insights [3,4], but these can only be retrospectively applied on activities already composed. Without support, activity composition requires expertise and case knowledge. Expertise is required, because one must be familiar with, for example, concepts of task and process design to ensure that activities are of the proper size, represent meaningful steps, and do not negatively affect execution efficiency. A modeler must furthermore be well-acquainted with the elementary information processing steps of a particular workflow. Due to the potential number of steps and possible complexity of their inter-relations, acquiring and applying the required case knowledge when designing activities can become a time-consuming task.

In this paper we set out to limit the expertise and domain knowledge that is required to properly compose workflow activities, whilst simultaneously reducing the time-intensiveness of this task. We achieve this by the provision of composition guidelines, that can be used to objectively construct *well-designed activities*. These guidelines, which extend a preliminary set introduced in [10], exploit the structural data-flow relations in a workflow. To make our ideas operational, we here capture such relations in a Product Data Model (PDM). However, the guidelines can be easily transferred to comparable data-flow specifications, such as the data flow matrices of [11]. The PDM that we build on stems from Product Based Workflow Design (PBWD), a methodology for the radical redesign of workflows [12]. We propose that data-flow relations, captured in the structure of a PDM, can be used to determine the *semantic relatedness* and *relative importance* of elementary information processing steps. These notions form the basis for the proposed composition guidelines, which result in activities that (i) are *meaningful* to workflow users executing them, (ii) form the basis for *understandable* process models, and (iii) do not result in *redundant* or *unnecessarily delayed* execution of tasks. Since the guidelines only consider the structural properties of a PDM, they can be applied without specific domain knowledge and can therefore be fully automated. This is demonstrated through an example implementation, which is freely available and can be used to automatically generate activities for any PDM. To illustrate the usefulness of our approach, we performed a thorough evaluation: We compare activity designs that have been automatically generated according to the propositions with designs that have been manually composed by experienced modelers.

The remainder of this paper is structured as follows. Section 2 introduces the PBWD methodology as well as a running example. The example illustrates the act of activity composition and its impact on workflow design. Section 3 describes the composition guidelines and illustrates how these can be applied in an automated fashion. Next, Section 4 shows the application of the proposed guidelines on a real-world business process. In Section 5, the performance of the guidelines is evaluated by comparing automatically generated activity designs with designs composed by experienced modelers. Section 6 discusses limitations of the current approach and directions for future research. Section 7 considers related work, after we conclude this paper with Section 8.

2. Activity composition

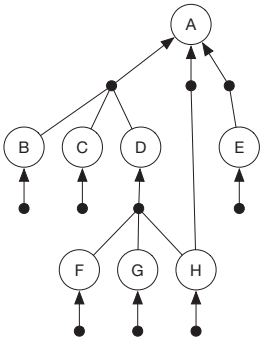
The composition guidelines are presented in this paper in the context of Product Based Workflow Design. This method is described in Section 2.1. Before the guidelines are proposed, we first provide a running example (Section 2.2) and demonstrate the impact that activity composition can have on the quality of process designs (Section 2.3).

2.1. Product Based Workflow Design

Product Based Workflow Design (PBWD) is a business process (re)design method [13]. It is one of the data-centric process (re)design methods that have emerged over the past decade to counter the overwhelming number of activity-oriented process modeling languages. Other well-known data-centric methods are *large process structures* by Müller et al. [14] and *artifact centric* process models introduced by Nigam and Caswell [15].

The PBWD method takes the informational product that is produced in the business process as the central concept. This product is built from several information processing steps, similar to how a Bill-of-Material (BoM) in [16] in manufacturing describes the composition of a physical product from its materials. The informational end product and its decomposition into sub data elements and input data elements is modeled in an hierarchical model, which is called the Product Data Model (PDM) [9]. This model describes the data elements and their relationships.

Fig. 1 contains a small and simple example of a PDM. It describes the calculation of the maximum amount of mortgage a bank is willing to provide to a client. The figure shows that there a three ways to compute the maximum mortgage amount (element *A* in Fig. 1). This amount is based on either (i) a previous mortgage offer (*E*), (ii) on the registration in the central credit register (*H*), or (iii) on the combination of the percentage of interest (*B*), the term of the mortgage (*C*), and the annual budget to be spent on the mortgage (*D*). The annual budget (*D*) is determined from the gross



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

Fig. 1. Small example of a Product Data Model describing the computation of a mortgage.

income of the client per year (*G*), the credit registration (*H*), and the percentage of the income the client is allowed to spend on paying the mortgage (*F*). The data elements of the PDM are depicted as circles. The *operations* on these data elements are represented by (hyper)arcs: the arcs are knotted together when the data elements are all needed to execute the particular operation. Compare, for instance, the arcs from *B*, *C*, and *D* leading to *A* on the one hand, to the arc from *H* leading to *A* on the other in Fig. 1. In the latter case, only one data element is needed to determine the outcome of the process (*A*), while in the case of *B*, *C*, and *D* all three elements are needed to produce *A*. These two operations also illustrate another important characteristic of a PDM. PDMs typically allow for a wide variety of alternative ways to generate the desirable end product. In this case, *A* may be produced by processing *B*, *C*, and *D*, or by processing *H* (under certain conditions, for instance that the value of *H* is 'negative'). This is in contrast to its manufacturing antipode, where the production process has fewer alternatives because of physical constraints.

When different operations can produce a value for the same data element, these operations are referred to as *alternative operations*. A distinguishable class of data elements is formed by the elements that represent the input to the workflow. The values for these elements, referred to as *leaf data elements*, are retrieved from outside a process, e.g. from a different process in the same organization, a client or from external parties [9]. Leaf data elements are produced by *leaf operations*, which are operations that do not require any input, such as the operations that create data elements *B*, *C*, and *E*.

Based on the PDM a process model can be derived. In the first PBWD case studies, this was performed manually [17], but later automatic algorithms were developed to generate process models based on a given PDM [18]. These algorithms translate the operations in the PDM one-on-one to activities in the process model, keeping the same level of granularity. The PDM, however, describes the information processing steps in a very detailed way. Often, the level of detail of the operations that transform input data elements to an intermediate output element is very high, which leads to small activities in the process that are hard to recognize as meaningful steps. In order to create logical pieces of work for workflow users (i.e. employees that execute the activities), researchers started investigating how to cluster or group several operations and their data elements into a meaningful activity. This resulted in the development of a first set of metrics for the assessment of a manually designed grouping [3]. These metrics, however, do not provide guidance on *how* to create the groups, nor were these metrics properly evaluated because of a lack of practical cases. This paper, therefore, introduces composition guidelines that support users when grouping operations into workflow activities. These guidelines are introduced in the context of a running example, as presented in Section 2.2.

2.2. Running example

This section presents a running example that is referenced in the remainder of this paper. The running example considers the governmental process that deals with requests for student grants in the Netherlands. Although the example, as introduced in [3], is based on a real world process, it is purposefully designed to introduce certain concepts in PBWD. The presented process is a simplified version of the actual procedure as implemented by the "Dienst Uitvoering Onderwijs", the governmental institution responsible for the assessment of student grant requests.

Fig. 2 depicts the PDM of the *student grants* example. Descriptions of its data elements are provided in Table 1. The goal of this workflow is to compute the total amount of student grant an applicant should receive. This value is captured in the root element: *i*42. Applicants that either do not have the Dutch nationality (*i*21) or are older than thirty years of age (*i*23) are not deemed eligible to receive a grant (*i*27). In those cases, the total amount of student grant assigned, straightforwardly, equals €0. Otherwise, the assigned amount forms the sum of four different sub-grants: (i) a basic grant (*i*40), (ii) a supplementary grant (*i*39), (iii) a loan (*i*41), and (iv) credit for tuition fees (*i*43). The basic grant is assigned to each applicant who is eligible to receive a student grant. The other three sub-grants must be specifically requested by an applicant. The eligibility and the amounts of these sub-grants depend on additional requirements, specific to each type of grant. Finally, it can be noted that all 11 leaf elements store information that is retrieved from a student's application. The interested reader is referred to [9] for a more detailed description of the students grant case.

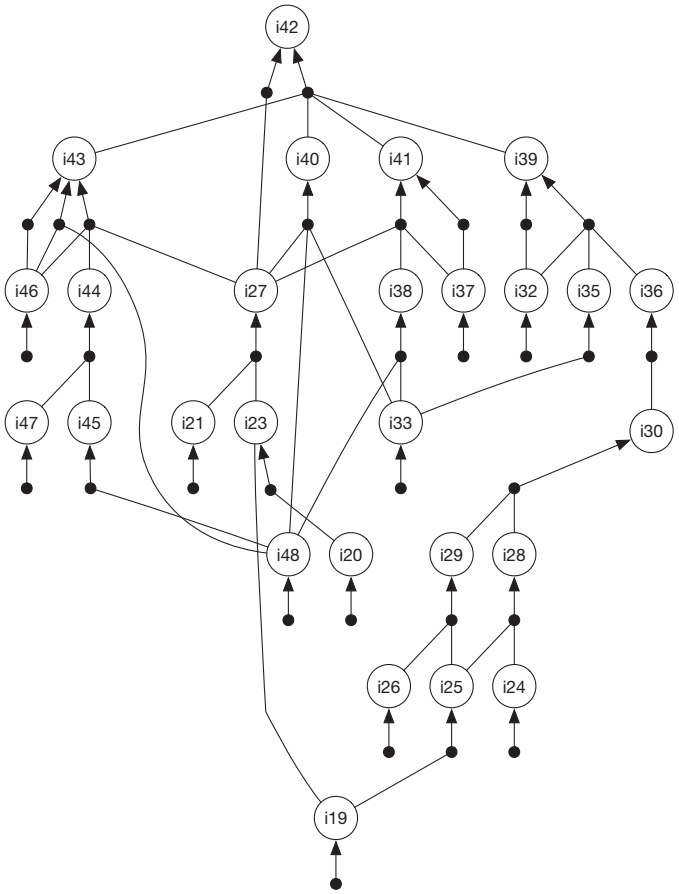


Fig. 2. Product Data Model of the student grants case.

2.3. Alternative designs

This sections demonstrates the impact that activity composition has on the execution and understandability of a workflow process. We present two different workflow process designs for the student grants example. The process designs are based on different sets of activities, i.e. *activity designs*. Both activity designs are considered *valid* according to [3], as they (i) include all data elements and operations of the PDM, and (ii) respect the dependencies of the PDM's informational structure. Nevertheless, the designs differ significantly in terms of their *execution efficiency*, *meaningfulness*, and *understandability*.

Table 1
Data elements in the student grants example.

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

2.3.1. Design 1

Fig. 3 presents a process model design for a preliminary set of activities. The process model contains six activities, of which the contents are depicted in Fig. 4. Activity *A* represents the receipt of a student's application. It consists of the 11 leaf operations and leaf data elements of the PDM. Activity *B* then uses the information derived from this application, i.e. the values stored in the data elements. The activity first determines an applicant's eligibility to receive a grant (*i*27). After which it continues with the calculation of the amount of basic grant assigned to an applicant (*i*40). After execution of activity *B*, the process model has two possible alternatives. First, if *i*27 is negative, the application is rejected by executing activity *H*, resulting in a value of zero for *i*42; the workflow process is thereby directly completed. The second alternative resembles acceptance of the application. In this alternative, the three remaining sub-grants are computed by activities *C* and *D*; these activities can be executed in parallel. Activity *C* determines the loan amount (*i*41) and the amount of tuition fee (*i*43) that are assigned to an applicant, while activity *D* determines the assigned amount of supplementary grant (*i*39). Based on the amounts for the four types of grant, activity *G* finally computes the total amount assigned to an applicant (*i*42).

Design 1 is a valid design that clearly illustrates distinguishable alternative paths for acceptance (activity *G*) and rejection (*H*) of an application. Nevertheless, the design has issues with respect to its execution efficiency and its understandability. First, activity *B* can be considered to be inefficient. *i*40, the value for the basic grant, is always computed, even when it is no longer required because the application will be rejected based on the value of *i*27. Any costs related to the computation of *i*40 would thus be spent unnecessarily. Furthermore, this activity design defers execution of activities *C* and *D* until after data element *i*40 is computed. This delay is not necessary, since these activities can be executed independent of element *i*40. A second issue involves the information that the process model provides about the underlying informational structure. Recall that a student grant consists of four different sub-grants. This information cannot be derived from Fig. 3, but only by analyzing the contents of the individual activities does it become apparent that there are four different sub-grants.

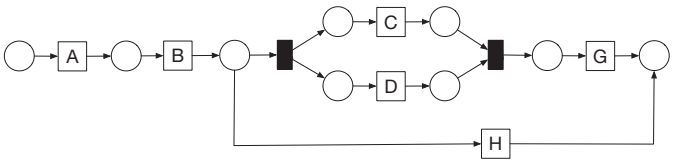


Fig. 3. Process model of alternative design 1.

The issues that are present in this design can be overcome by applying a small number of adjustments to 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 *B*1 and *B*2, and *C*1 and *C*2. The resultant process model is shown in Figs. 5 and 6. This process model clearly resembles the core structure of the PDM. Activities *B*2, *C*1, *C*2, and *D* clearly illustrate that there are four sub-grants that can be independently computed after an application is accepted. Furthermore, this design overcomes the aforementioned efficiency issues that are present in design 1. Due to the division of activity *B* into two separate activities, execution of other activities is no longer unnecessarily deferred. The final important improvement is related to the semantics 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 seemingly unrelated sub-grants. The activities in design 2, by contrast, all consider a single, distinguishable part of the process. Furthermore, each activity results in the computation of a data element that represents a relatively important intermediate result of the workflow. Hence, each activity in the second design represents a logical step towards the desired end result.

The two alternative process designs illustrate the impact that well-reasoned activity composition can have on meaningfulness of activities, and process understandability, and execution efficiency.[1] However, this leaves open the issue of how to approach the task of activity composition in a structured way. This is addressed in Section 3.

3. Composition guidelines

The second alternative design considered in the previous section illustrates that proper activity composition can have on meaningfulness, understandability and execution efficiency. This section presents three composition guidelines that can be used to construct such proper activity designs.

This sections presents the composition guidelines we propose that can be used to create well-designed activities. The guidelines are based on the proposition that certain semantic aspects of a workflow can be derived from structural properties captured in a

[1] Note that while in this particular case the process steps can be automated and the effect on execution efficiency would be limited in reality, we later on demonstrate that these effects are transferable to processes that do include significant human involvement.

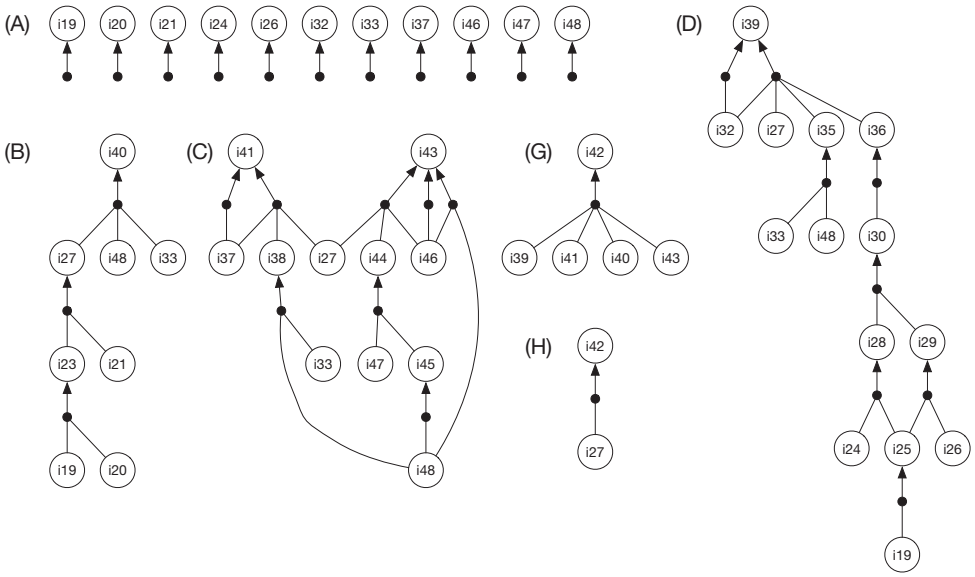


Fig. 4. Activities of alternative design 1.

PDM. These semantic aspects comprise: (i) the importance of certain data elements, and (ii) the semantic relatedness of operations. The composition guidelines that utilize the proposed relations are purposefully designed to be applicable independent of context or any information beyond the structure of a PDM. The proposed guidelines result in activities that work towards the production of an important data element on the one hand and of which the operations are semantically related on the other.

This section comprises three parts. Section 3.1 proposes a relation between the importance of data elements and structural patterns in a PDM. Second, Section 3.2 considers the identification of semantic relatedness based on structural properties. The propositions introduced in these sections represent composition guidelines that can be used to create activity designs that are proposed to consist of well-designed activities. Since the guidelines can be applied in an objective fashion, it is possible to generate activity designs in a fully automated fashion. An algorithm that achieves this automated generation, as well as a freely available implementation are presented in Section 3.3.

3.1. Data element importance

Although by definition all data elements in a PDM are required to achieve the desired end result of a workflow, it is clear that in general not all data elements are of equal importance. For example in the student grants case, data element *i*27, an applicant's eligibility to receive a grant, is certainly more important than *i*28, the income of the applicant's father. While the latter data element only indirectly influences one out of the four sub-grants which an applicant may receive, the former data element directly determines whether an applicant receives any financial support or not. Our first proposition, described in this section, focuses on those data elements that are important in a PDM.

We aim to identify these elements in order to be able to design activities that work towards the computation of an important data elements. Thereby, the activity designs adhere to notions of *task identity* and *task significance*, two determinants of *experienced meaningfulness* in the widely used Job Characteristics Model of Hackman and Oldman [5]. Task identity is the degree to which a job (i.e. a single activity) 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 designing activities such that they work towards a specific goal is also regarded as a best practice in (electronic) form design (see e.g. [19]). Form design is arguably closely related to the execution of activities by users when a process is supported by *workflow applications* and a *workflow management system*; designing activities such that they meet best practices in form design is thus desirable.

The relative importance of data elements can sometimes be inferred from their descriptions, as is the case for data element *i*27. Assessing the importance of a data element in this manner, i.e. from a semantic perspective, requires domain knowledge and depends on subjective judgment. Modelers that are unfamiliar with the process thus face a difficult and time consuming task when manually selecting important data elements. To avoid this subjective task, this section introduces five structural patterns that occur in PDMs which are proposed to reveal important data elements (IDEs). This concept is based on the proposition that the structure of a PDM is a crucial determinant when evaluating the importance of data elements. The five patterns have been found by analyzing manual activity designs from [3,9,17] and by considering best modeling practices in the context of PBWD.

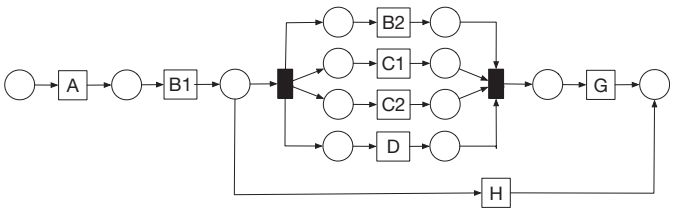


Fig. 5. Process model of alternative design 2.

Pattern 1 (Root data element).
The root data element is the single data element that is not used as input for any operations.

The root data element represents the desired end result of a workflow and is therefore straightforwardly the most important element in a PDM. For example, in the student grants case, the root

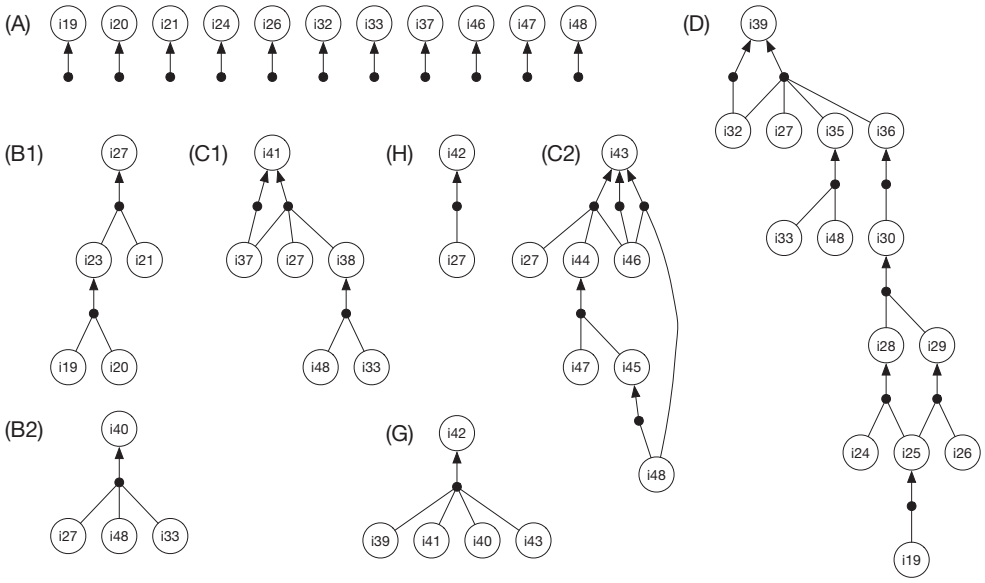


Fig. 6. Activities of alternative design 2.

element *i*42 represents the total amount of student grant assigned to an applicant. Although this first pattern is trivial, explicitly marking the root element as an IDE is important in the remainder of the proposed approach.

Pattern 2 (Leaf data elements).
A leaf data element is a data element that is produced by an operation without input elements.

The second type of IDEs that can be distinguished based on structural properties are *leaf data elements*. These are the pieces of information that are provided as input to a workflow; the values for these elements are retrieved from outside a process, e.g. from a different process in the same organization, a client or from external parties [9]. In the student grants example, the leaf data elements represent the information that is provided by the applicant, e.g. *i*20 the birth date of the applicant and *i*37 the requested amount of student loans. Leaf data elements are here considered as important, since they are clearly different from data elements computed inside the workflow.

Pattern 3 (Conditional data elements).
A conditional data element is a data element that can be produced by multiple alternative operations.

The third type of IDEs are *conditional data elements*. These data elements can be produced by multiple, alternative operations. Often, the choice for a particular alternative operation depends on the values of previously produced data elements and on certain *execution conditions*. E.g. in the fragment of student grants example depicted in Fig. 7, the left-hand operation to produce data element *i*39 can only be executed if the applicant does *not* request a supplementary grant. Otherwise, the execution conditions force the workflow to execute the right alternative operation, which requires a larger set of input elements. Conditional data elements are regarded as important based on the premise that they represent an intermediate result of the workflow, since they are the outcome of a decision in the process. This is for example clear in the student grants case, where, aside from the root element *i*42, the other conditional data elements *i*39, *i*41, and *i*43 represent the amounts assigned to an applicant for three of the sub-grants that underlie the total student grant amount. These sub-grants are without a doubt important, since together they directly determine the outcome of the workflow process.

Pattern 4 (Equal level data elements).
An equal level data element is an input data element to an operation that also requires at least one conditional data element as input.

Pattern 3 identifies three out of the four sub-grants as IDEs in the form of conditional data elements. Without contextual information, there is no reason why the fourth sub-grant, *i*40 the amount of basic grant assigned to an applicant, is less important than the other sub-grants, despite that it is not a conditional data element. Pattern 4, therefore, introduces *equal level data elements*. These are input elements to an operation which also requires conditional data elements. Equal level data elements are based on the idea that all input used for such operations is important. This is for example seen in Fig. 7, which shows a fragment of the student grants case related to data element *i*40. The operation that produces *i*42 in this figure combines the values for all four sub-grants in order to determine a value for the total amount of student grant.

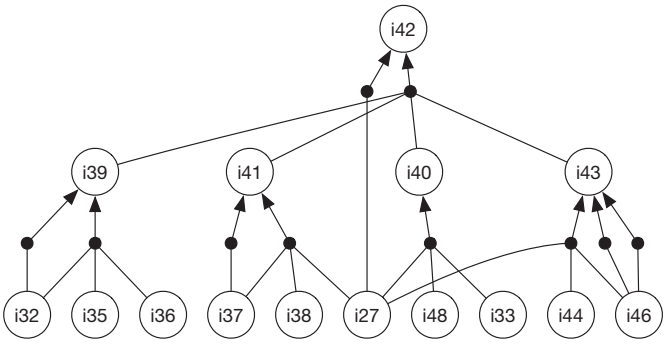


Fig. 7. Equal level data element *i*40 in the student grants case.

Pattern 5 (Reference data elements).
A reference data element is a data element that is an input element to two or more operations that are each, directly or indirectly, involved in the computation of different sets of data elements that match one of the four previous patterns.

Recall that *i*27, the eligibility to receive a grant, is arguably an important data element from a semantic point of view, since it determines whether or not an applicant receives any student grant at all. This data element is, however, also important from a structural perspective, because it is required as input for five operations. Pattern 5 identifies such data elements that are used as input to a variety of operations. A data element is considered to be a *reference data element* if it is an input element to multiple operations that are, directly or indirectly, involved in the computation of different IDEs. This is, for example, the case for *i*27, which is required as input to operations involved in the computation of five different IDEs: *i*39, *i*40, *i*41, *i*42, and *i*43. Pattern 5 excludes data elements that are required as input elements for multiple operations, when these operations are all involved in the computation of a value for the same IDE. Such elements are not considered to be as important, since their value is only relevant to a single IDE, while reference data elements are relevant throughout a larger part of the workflow. E.g. in the student grants case, element *i*37, the requested loan amount, is only relevant for IDE *i*41, the amount of loan assigned to an applicant, even though *i*37 is an input element to multiple operations.

The notion that the five introduced patterns identify important data elements on the basis of structural properties is captured in Proposition 1.

Proposition 1 (Important data elements).
Root, leaf, conditional, equal level, and reference data elements represent important data elements in a PDM.

3.2. Semantic relatedness

Semantic relatedness considers the degree to which the meaning of things is alike. In a PDM, this is most notably found in data elements that are related to a similar topic or an object and the operations that produce those. The description of data elements can be an important determinant for this relatedness. A clear example follows from the descriptions of the data elements in Table 2. The eight elements can be easily separated into two groups based on their description. The former three elements, *i*37, *i*38, and *i*41, are each clearly related to *loans*, while the latter five elements all deal with *tuition fees*.

Similar to the propositions made regarding data element importance in Section 3.1, we propose that there also exists a relation between semantic relatedness and the structure of a PDM. By exploiting this relation, it is possible to objectively identify semantic relatedness between operations without requiring contextual information or linguistic analysis.

Fig. 8 illustrates the intuition that underlies this proposition. The figure shows the operations that produce the eight data elements considered in Table 2 and their interrelations in the PDM. It can be observed that those five operations that produce a data element related to tuition fees, marked by the right dashed area, have a path towards *i*43, the amount of tuition fee credit assigned to an applicant. The same holds for the three operations that produce data elements related to student loans, that have a path to *i*41. By contrast, *i*27, an applicant's eligibility to receive a grant, and *i*48, the type of education of an applicant, are both semantically related to neither student loans nor to tuition fees, while they have paths to both operations. The operations that produce these data elements should therefore not be included in the same groups as the operations related to the either of the sub-grants.

Table 2
Data elements with clear semantic relatedness.

ID	Description
i37	Requested amount of loan
i38	Maximum amount of loan
i41	Amount of loan assigned to applicant
i43	Amount of tuition fee credit assigned to applicant
i44	Maximum amount of tuition fee credit
i45	Tuition fees of educational institution
i46	Applicant has requested credit for tuition fees
i47	Tuition fees declared by law

The second composition guideline associates each operation with an IDE, such that those operations that are related to the same IDE are considered to be semantically related to each other. The five operations in Fig. 8 that are related to tuition fees are, for example, associated with data element *i*43. This IDE is referred to as the *associated IDE* of those operations. For each operation, there exists exactly one associated IDE in a PDM according to Definition 1.

Definition 1 (Associated IDE).
The associated IDE of an operation is the unique IDE for which there exists a path in the PDM from the operation to that IDE, such that this path does not contain any other IDEs.

The property that each operation has exactly one associated IDE results from the definition of reference data elements. Every operation that would otherwise have uninterrupted paths (i.e. not containing an IDE) to multiple IDEs, by definition either produces a reference data element or has a reference data element in the paths towards the other IDEs. Since a reference data element is also recognized as an IDE, an operation can thus never be associated with more than one IDE. A formal proof of this property is provided in [20]. By grouping together operations that have the same associated IDE, activities are constructed of which the underlying operations are related to a similar topic, i.e. semantically coherent activities. Two special cases extend this notion of semantic relatedness: leaf activities and alternative paths.

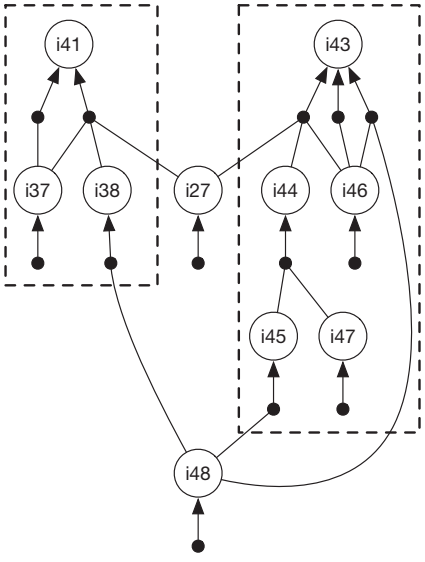


Fig. 8. Identification of semantic relatedness through structural properties.

First, it is proposed that a semantically coherent activity can also be composed by grouping leaf operations into a *leaf activity*. This practice is found in several manual activity designs [see e.g. 3, 9, 17]. This notion is based on the premise that the values for multiple leaf elements are often retrieved from the same information source and thus have a certain relatedness. For example, as seen in Section 2.2, all leaf elements in the student grants case are derived from a student's application. By grouping together leaf operations, the workflow design enforces the retrieval of multiple data elements at the same time. This is often desirable since it reduces the number of contacts with customers and third parties [21]. For the sake of conciseness, it is said that leaf operations are all associated to the same IDE.

Second, we set out to keep important decision logic visible in the process model, rather than hidden inside an activity. Namely, we wish to avoid composing activities that contain multiple alternative operations that depend on the execution of different sets of activities. This ensures that any process model, containing activities designed according to the proposed guidelines, clearly visualizes the most important alternative execution paths. As an illustration, consider the two operations that can produce the root element in the running example. If an activity contains both of these operations, it obscures the fact that the process has two distinct outcomes: namely rejection and acceptance of an application. By contrast, if we create separate activities for both of the operations, the resulting process model clearly shows that there are two different results possible. The second design for the running example, depicted in Fig. 9, illustrates this. Arguably, this process model provides a more accurate representation of the information structure underlying the process than a model with just a single end activity. Note that, while for the case of two alternative execution paths, the difference can be considered relatively small, this quickly changes when the number of execution paths increases.

Proposition 2 (Semantically coherent activities).
A semantically coherent activity is an activity that consists of operations that have the same associated IDE and does not obscure important decision logic.

Finally, it is proposed that a well-designed activity works towards the production of an important data element and consists of semantically coherent activities. By combining the former two propositions, it is possible to compose such activities in an objective manner, purely based on the structure of a PDM. This is defined in Proposition 3.

Proposition 3 (Well-designed activities).
Well-designed activities consist of semantically coherent activities that work towards the production of an important data element.

Note that by applying the presented composition guidelines on the running example, the exact same activity design is reached as presented in Section 2.3.2. This design was shown to result in an efficient and understandable process model that contains activities that each represent a logical step in the workflow. After describing an implementation of the guidelines in Section 3.3, Section 4 demonstrates the application of the guidelines on a different, real life business process, whereas Section 5 compares generated activity designs to designs created by experienced modelers.

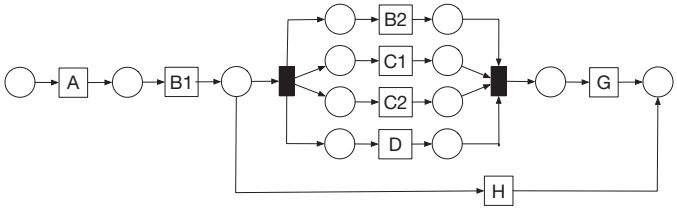


Fig. 9. Process model of alternative design 2.

3.3. Implementation

The proposed activity composition guidelines are designed to be objective and independent of information beyond the definition of a PDM. This means that when a PDM is represented in a formal manner, e.g. as a directed graph or in an XML-file, the composition guidelines can be applied without input from a user. This thus results in the automated generation of a set of activities for any given PDM.

This section describes the construction of an algorithm that implements the composition guidelines. The implementation is intended as a proof of concept to demonstrate that the guidelines can be automated. For this reason, the described algorithm favors understandability over computational efficiency. It consists of three subsequent steps. First, IDEs are identified according to the patterns presented in Section 3.1. Second, operations are associated to the previously identified IDEs, such that groups of semantically related operations are formed. Finally, those groups of operations that would obscure important decision logic when translated into a process model are split into multiple sets. The resulting sets of operations each represent an activity that is in accordance with the proposed activity composition guidelines.

3.3.1. Identifying important data elements

The identification of data elements that match one of the four first patterns, i.e. root, leaf, conditional, and equal-level data elements, is straightforward. Algorithm 1 matches data elements to these patterns based on a node v's predecessors, denoted as *predecessors(v)*, and successors (*successors(v)*).

Algorithm 1. Identification of IDEs.

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

A reference data element is defined in Pattern 5 as a data element that is an input element to two or more operations that are each, directly or indirectly, involved in the computation of different sets of IDEs (i.e. data elements matching one of the first four patterns). This definition reveals that the identification of reference data elements is less straightforward than that of the other four patterns. First, because the pattern requires other IDEs to have been previously identified. Second, the definition of the pattern states that indirect connections between elements must

also be considered, thus it does not suffice to consider only the predecessors and successors of a data element.

An operation o is here said to be *involved in* the computation of a data element d if there exists a path in the PDM from the operation to that data element, i.e. d is reachable from o. Reachability between elements is determined by computing the transitive closure. The set of vertices in a PDM that is reachable from a vertice v is is denoted as *reachable(v)*.

In order to identify reference data elements, Algorithm 2 thus searches for data elements that have two or more operations in their successors, such that these operations are able to reach differing sets of IDEs (line 9).

Algorithm 2. Identification reference data elements.

```
1:   input1: PDM
2:   input2: IDEs ▷ Data elements identified by Algorithm 1
3:   newIDEs = ∅
4:   for d ∈ PDM.dataElements do
5:     for op1 ∈ successors(d) do
6:       op1ReachableIDEs = reachable(op1) ∩ IDEs
7:       for op2 ∈ successors(d) do
8:         op2ReachableIDEs = reachable(op2) ∩ IDEs
9:         if op1ReachableIDEs ≠ op2ReachableIDEs then
10:          newIDEs = newIDEs ∪ {d}
11:  return newIDEs
```

3.3.2. Determining associated IDEs

Definition 1 states that the associated IDE of an operation is the single IDE for which there exists a path from that operation to the IDE, such that the path contains no other IDE. This means that each IDE has a set of (one or more) operations that have a path towards this IDE, without the path containing a second IDE. This is the set of operations that are associated to that IDE. These sets are easily identified by considering the prerequisites of IDEs, i.e. all operations and data elements that have a path towards an IDE, as shown in Algorithm 3. To compute the operations associated with an IDE d_1, first include all prerequisite operations of d_1 as candidates (line 5). Then, for each different IDE d_2 that does not require d_1 as a prerequisite (line 7), remove the prerequisites of d_2 from the candidate operations (line 8). Note that the requirement defined in line 7 is crucial here, since this ensures that only the operations that have an uninterrupted path (i.e. not containing other IDEs) towards d_1 are associated with that IDE. After removing the prerequisites of all relevant IDEs from the list of candidate operations, the remaining operations are the associated operations of d_1.

Algorithm 3. Determining associated operations.

```
1:   input1: PDM
2:   input2: IDEs
3:   activities = []
4:   for d1 ∈ IDEs do
5:     activity = prerequisites(d1)
6:     for d2 ∈ IDEs do
7:       if d1 ≠ d2 and d1 ∉ prerequisites(d2) then
8:         activity = activity − prerequisites(d2)
9:         activities = activities ∪ {activity}
10:  return activities
```

3.3.3. Maintaining decision logic

Proposition 2 concludes with the notion that a semantically coherent activity does not obscure important decision logic. Specifically, we wish to avoid composing activities that contain multiple alternative operations, when these operations require the prior execution of different sets of activities. Since, by definition, each activity produces exactly one IDE, we identify these situations by considering the sets of IDEs required to execute alternative operations. Algorithm 4 identifies these cases in a straightforward manner. The activities that are identified by Algorithm 4 must be split into sets of operations that each contain only one of the alternative operations. After completion of this step, an activity design has been created in which the sets of operations all represent well-designed activities, according to Proposition 3.

Algorithm 4. Identification of important decision logic.

```
1:   input1: PDM
2:   input2: IDEs ▷ Data elements identified by Algorithms 1 and 2
3:   input3: activities ▷ Sets of operations computed by Algorithm 3
4:   result = ∅
5:   for d ∈ goals do
6:     for altop1 ∈ predecessors(d) do
7:       for altop2 ∈ predecessors(d) do
8:         difference = prereqs(altop1) − prereqs(altop2)
9:         if {difference} ∩ {IDEs} ≠∅ then
10:          result = result ∪ {d}
11:  return result
```

3.4. Tool support

The algorithm described in Section 3.3 has been implemented as a *plug-in* for the ProM6-framework.[2] ProM6 is an extensible framework focused on process mining, which includes support for PDMs. 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 Fig. 10.

The plug-in allows modelers to create activity designs in two ways. First, users can fully automatically generate a set of activities that adheres to the guidelines of Section 3. Second, since we recognize that in some cases activities are designed by incorporating information beyond the structural properties of a PDM, the plug-in allows users to customize activity designs by manually selecting data elements that the users considers to be important. This set of elements is then used to construct an activity design in which the manually specified data elements are used as output elements for the activities. Furthermore, since the generation of activities takes mere seconds, the tool is ideal for iterative refinement. A modeler can generate activities until a design best meets his preferences.

After an activity design has been created, users are provided with two types of results. First and foremost, the individual activities of the activity design can be inspected. Second, the implementation produces an abstracted version of the original PDM. In this version, each activity replaces its set of underlying

[2] http://www.promtools.org/prom6.

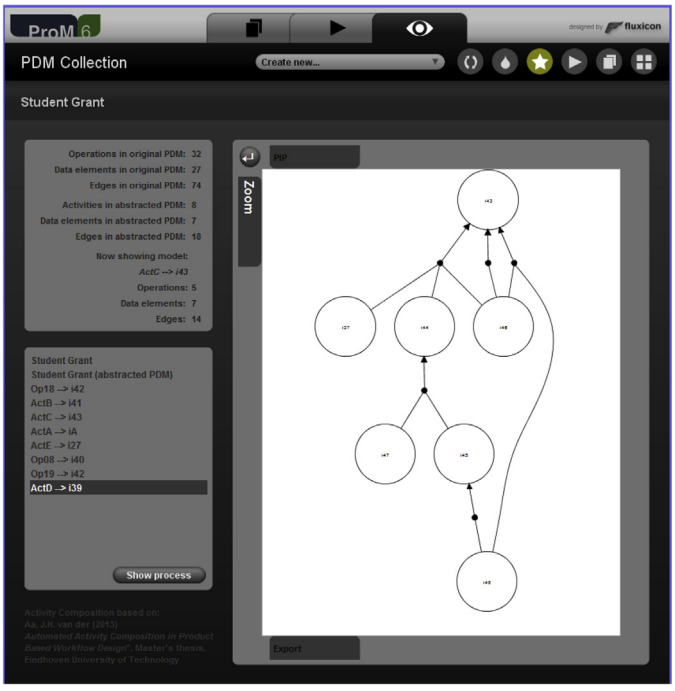


Fig. 10. Screenshot of the ProM6-plugin.

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. Fig. 11 shows the abstracted PDM of the student grants case. It clearly shows different paths to compute the root element (representing acceptance and rejection), and illustrates that the former option requires the computation of four sub-grants. The abstracted PDM can furthermore form the basis for the design of process models, either manually or by applying process generation algorithms (see e.g. [18]).

Due to the combination of automated support and customization, the ProM6 plug-in is a means that allows users to quickly generate, view and customize activity designs for a given PDM. It presents the first available tool to support activity composition for workflow processes.

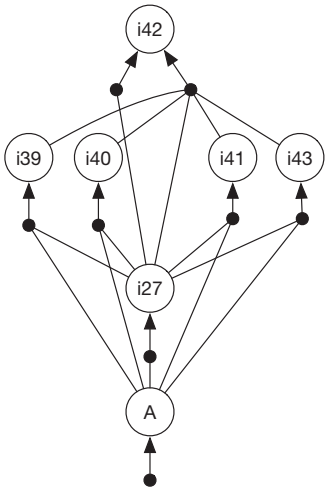


Fig. 11. Abstracted PDM of the student grants case.

4. Application

Section 3 introduced guidelines that can be used to automatically compose well-designed workflow activities. This section presents an application of the guidelines on a second, real life business process. Section 4.1 introduces the fireworks case used in this application. Section 4.2 presents the activity design and an accompanying process model that follow from the application of the guidelines on the PDM. Finally, Section 4.3 discusses points of improvement in the generated design by comparing it with an activity design created by a human modeler.

4.1. Case description

The PDM used in this application is based on an actual procedure as implemented by the "Provincie Noord-Brabant" (PNB), which is one of the 12 Provincial Councils in the Netherlands. The responsibilities of PNB include the granting of subsidies, permissions and licenses with respect to environmental policies. The process under consideration assesses requests for permission to ignite fireworks during special events. Such events, for example, concern fairs, theatrical plays or musical performance. Although no execution times are known for this process, the description of the process clearly shows that the workflow involves a significant amount of human work.

The PDM of the *fireworks* case is shown in Fig. 12; it contains 81 operations and 46 data elements. The PDM was originally created during a redesign project described in [9]. The final outcome of the workflow is a decision on whether or not to grant permission to a request. This result is captured in the root element of the PDM: *i*01. The workflow takes 28 leaf elements as input, which are retrieved from an application form. The contents of these leaf elements range from information on the time and location of the event, to the relevant insurance policies of the applicant. A description of the leaf and all other data elements is provided in Table 3.

The 28 leaf elements are used to determine five intermediate assessment results: (i) acceptance of the applicant (*i*12), (ii) validation of insurance (*i*13), (iii) acceptance of the request (*i*14), (iv) acceptance of the list of firework pieces (*i*15), and (v) acceptance of the location. Based on the outcome of these five assessments, a draft of a decision is drawn (*i*10). This conceptual decision is then sent to a number of external parties with expert knowledge, e.g. the fire brigade, the mayor, and the Department of Waterways and Public Works. These parties are requested to assess the concept decision and give their advice or statement of no objection. Because it is decided per case which external parties should give their approval of the conceptual decision, there are many alternative operations to produce data element *i*02: the final decision. This decision is finally communicated to the applicant as the fixed final decision (*i*01).

4.2. Generated activity design

The ProM6 tool, described in Section 3.3, is used to generate an activity design for the fireworks case. The nine activities that comprise this design are depicted in Fig. 13. For reasons of brevity, the size of activity A is reduced by omitting a majority of the leaf elements. A process model based on this activity design is presented in Fig. 14.

In accordance with the notion to group leaf operations together, activity *A* contains all those 28 operations, i.e. those operations that produce data elements derived from the application form. Activity *B* represents a preprocessing step that derives the exact location of the event (*i*40) based on the provided address (*i*41) and a map of the environment (*i*42). This preprocessing step is included because

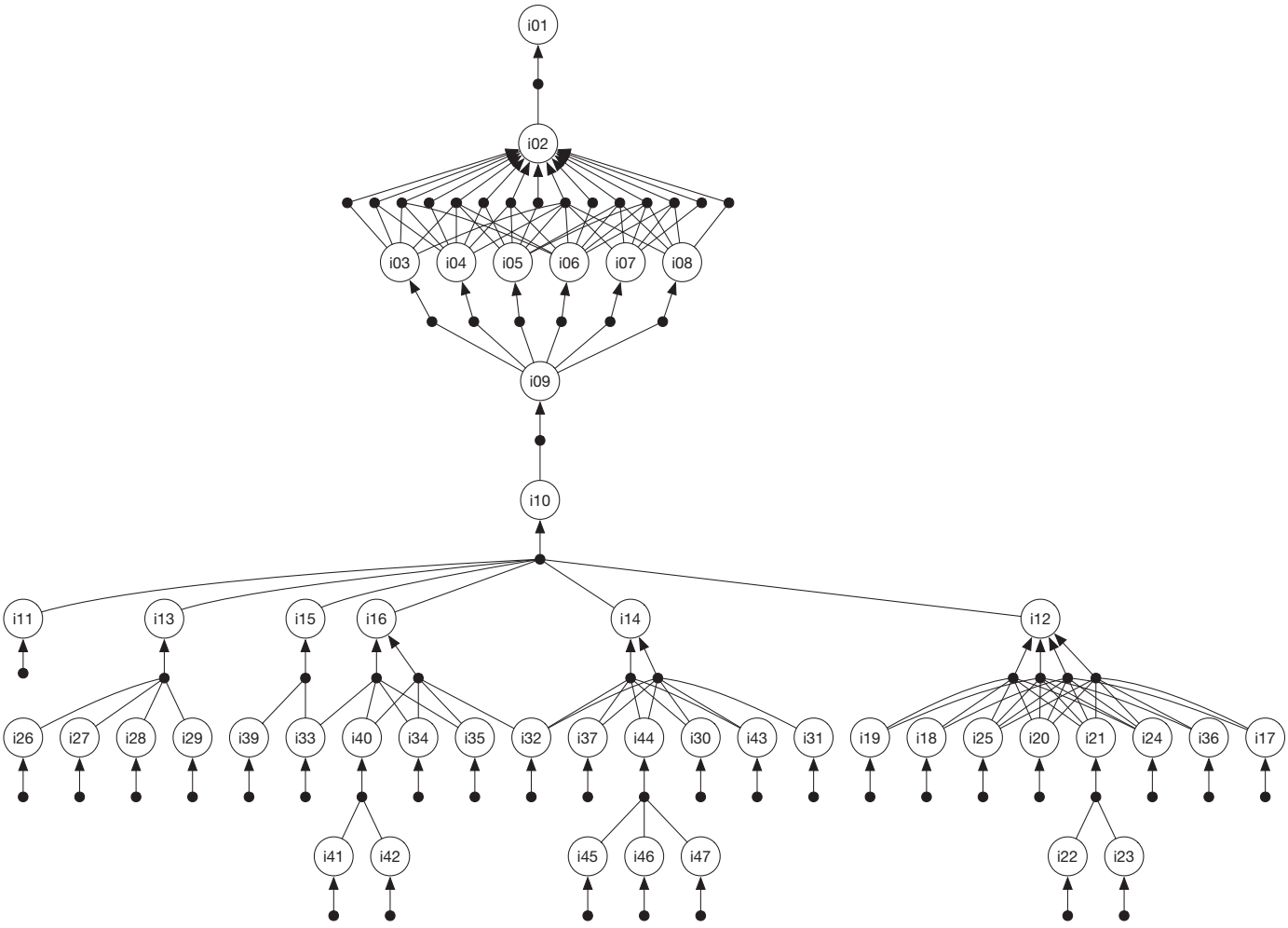


Fig. 12. Product Data Model of the fireworks case.

element *i*40 is required as input to two of the other activities, i.e. it is a reference data element. After this preprocessing step, the data values computed in activity *A* and *B* are used as input for the five assessment activities: activities *C*1 to *C*5. Each of these activities takes between two and nine input elements and produces a value for one of the intermediate assessment results (*i*11–*i*16). For example, activity *C*1 assesses an applicant's eligibility to receive a permission based on, amongst others, an applicant's license and whether or not the application is in possession of a certificate of competence.

After the intermediate assessments have been conducted, activity *D* comprises the steps that draft a conceptual decision up to the final decision, based on the judgment of external parties. Finally, activity *E* concludes the workflow by communicating the fixed decision to the applicant.

4.3. Analysis

The activity design of Section 4.2 arguably shows the most important characteristics of the Fireworks workflow. For example, the design clearly shows that five intermediate assessments must be performed (and can be done so in parallel) before a decision can be made. Nevertheless, it is conceivable that a human modeler would make a different design decision. The case description notes that first a conceptual decision is made before external parties are included in the process, which ultimately leads to a final decision. The generated activity design captures all these parts in activity *D*. However, if we look beyond just the structure of the PDM, it is probably a better choice to split this activity such that the conceptual and final decisions are determined by different activities. This design choice is also made in the manual activity design presented in [9]. Splitting this activity emphasizes the existence of two different decisions made.

The noted difference exemplifies how generated and manual activity designs can differ due to implicit information (i.e. the required involvement of external actors) that is at the disposal of and utilized by human modelers, but not considered in the composition guidelines. The evaluation in Section 5 sets out to quantify such differences between generated and manual activity designs in order to determine how well the composition guidelines approximate design choices made by experienced modelers.

5. Evaluation

The proposed activity composition guidelines call for evaluation. The goal of this quantitative evaluation is to determine how well the guidelines approximate design decisions made by modeling experts. To determine this, the designs that are automatically generated according to the guidelines presented in this paper are compared to designs that have been created manually by experienced modelers.

Table 3
Data elements in the fireworks case.

ID	Description	ID	Description
i01	Fixed final decision on the firework ignition permission	i24	Firework licence of applicant
i02	Final decision on the firework ignition permission	i25	Certificate of competence
i03	Statement of no objection of mayor (M)	i26	Copy of valid insurance policy
i04	Advice of the fire department (FD)	i27	Duration of insurance
i05	Advice of the Provincial Executive (PE)	i28	Insurance company
i06	Advice of the labor inspectorate (LI)	i29	Insurance policy number
i07	Advice of the Department of Waterways and Public Works	i30	Agreements parties concerned
i08	Advice of the air force (AF)	i31	Additional information
i09	Fixed concept of decision	i32	Table with work, dangers and measures
i10	Concept of decision	i33	List of firework pieces
i11	Standard regulations	i34	Additional information
i12	Acceptance of applicant	i35	Visit on location
i13	Validation of insurance	i36	KIWA site
i14	Acceptance of request	i37	Contact information of the organization
i15	Acceptance of the list of firework pieces	i39	VROM table
i16	Acceptance of location	i40	Location
i17	List of holders of a firework licence	i42	Map of the environment of the event
i18	List of holders of a certificate of competence	i43	Kind of event
i19	Authority that has granted the firework licence	i44	Date and time of event
i20	Contact addresses	i45	Start time building and construction work
i21	Company	i46	Start time ignition of firework
i22	Address of the company	i47	End time ignition of firework
i23	Post box number of the company		

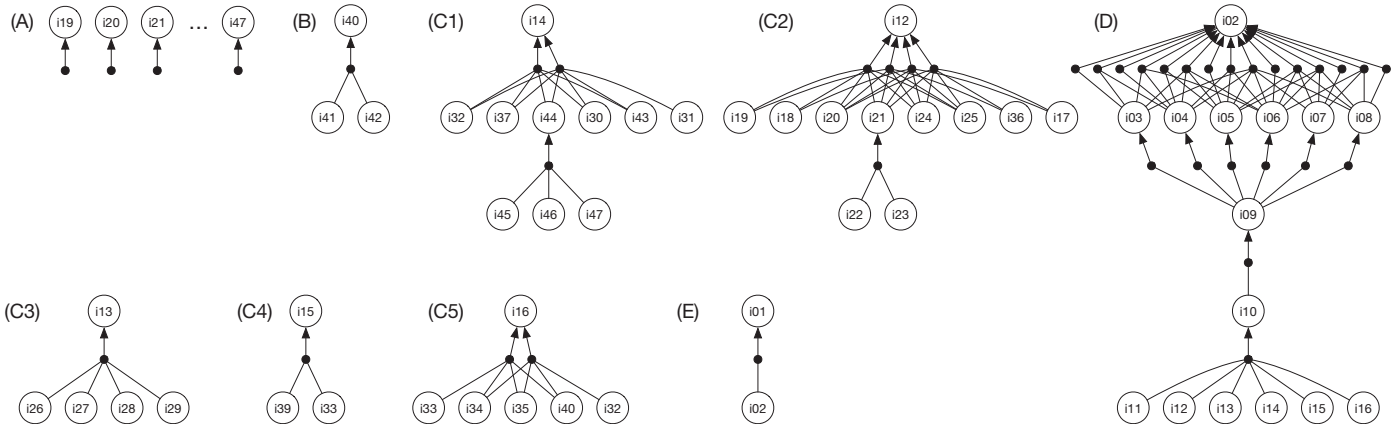


Fig. 13. Generated activity design of the fireworks case.

5.1. Setting

The evaluation compares automatically generated activity designs, referred to as *proposed designs*, with manually created activity designs, referred to as *solutions*. Five cases have been selected from the literature. The student grant case (SGNL) of [3] and the fireworks case (FWRK) of [9] have already been introduced. The other three cases are: the bicycle manufacturing case (BICL) of [22], the unemployment benefits case (UNEM) of [17], and a second student grants case (SGUS), based on the application process in the United States, described in [20].

With the exception of the BICL case, four of the five cases represent real-world processes. The PDMs of the cases range from a PDM with 32 operations and 27 data elements (SGNL), to a PDM with 81 operations and 46 data elements (FWRK). In total there are 11 solutions available for the five cases.[3] All solutions have been designed by modelers that are both well-versed in PBWD and have an expertise in the case's domain. The solutions represent designs that emphasize various aspects that can be considered during activity composition. For example, SGNL solutions focus on the correct size of activities and therefore mainly differ in their level of granularity; the solutions for the UNEM case have been designed to optimize execution efficiency and therefore carefully consider the processing times and cost of operations; and the resource or organizational perspective plays a leading role in the designs for the BICL case. The 11 available solutions, therefore, comprise a varied spectrum of design approaches. Extended information on these solutions is available in [20].

Recall from Section 3 that the SGNL and UNEM cases (as well as their solutions) have been analyzed during the design of the propositions. To maintain objectivity, the evaluation results of these cases are hence separately presented as *training set* results. The three previously unused cases, consequently, form the *validation set*.

5.2. Proposition 1

Proposition 1 states that important data elements in a PDM can be identified based on five structural patterns. This proposition is evaluated by comparing the data elements that match the proposed patterns with the output elements of activities in the

[3] The number of available solutions per case is: SGNL (3), UNEM (2), BICL (2), FWRK (1), SGUS (3).

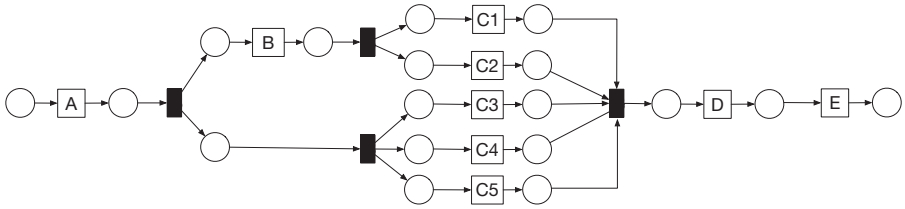


Fig. 14. Process model of activity design for fireworks case.

solutions. The validation is thus based on the assumption that these solutions, similar to the proposed designs, are designed such that the an activity produces a value for an IDE. This assumption appears to be valid for all solutions, except for those of the UNEM case. The description of the solutions in [17] explicitly states that these were designed based on execution efficiency, while excluding semantics. For this reason, the UNEM case is not considered in the validation of Proposition 1.

5.2.1. Metrics

Proposition 1 is evaluated by comparing the output elements of activities in the proposed designs with the output elements of activities in the solutions. The comparison is quantified by computing *precision*, *recall*, and *F-score*. Precision indicates the fraction of IDEs in the proposed designs that match output elements in the solutions, whereas recall is the fraction of output elements in the solutions that are also present in the proposed designs. Finally, the F-score provides the harmonic mean of precision and recall [23].

5.2.2. Results

The average results per case are presented in Table 4. The scores on all three seem high, with notable high averages for precision of 1.00 (training set) and 0.87 (validation set). This implies that the patterns are valuable predictors of output elements in the solutions, thus according to our assumption, for important data elements. 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 (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.

5.3. Proposition 2

Proposition 2 states that a semantically coherent activity consists of operations that have the same associated IDE. To validate whether these generated activities are indeed semantically coherent, it is assumed that the activities in the solutions also consist of semantically related operations. As with the evaluation of Proposition 1, this assumption once again results in the exclusion of the UNEM case, since these solutions explicitly do not consider semantics.

5.3.1. Metrics

Proposition 2 is evaluated by comparing the contents of activities, i.e. their operations, in the proposed designs with those of activities in the solutions. In order to evaluate Proposition 2, independent of the correctness of Proposition 1, the proposed designs are generated by selecting the output elements of the activities in the solutions as IDEs, rather than using the proposed patterns to identify these.

Each activity represents a *cluster* of operations. Therefore, to assess the similarity between activities in the proposed designs and the solutions, two cluster performance metrics are utilized: the *Rand index* [24] and *Jaccard index* [25]. For each pair of operations, both indices determine whether or not these operations are part of the same activity in the proposed design and the solution. The Rand index calculates the fraction of both *true positives* and *true negatives*, i.e. correctly identified similarity as well as dissimilarity. By contrast, the Jaccard index only focuses on correctly identified similarity. The latter metric is important for the evaluation of fine-grained solutions, where a high number of true negatives may conceal a low number of true positives. The possible values of both indices range from 0.00 to 1.00, where the latter indicates perfect similarity between a proposed design and a solution.

5.3.2. Results

Table 5 shows that the average scores of the Jaccard and Rand indices are, respectively, close to and well-above 0.90. This indicates that the automatically generated designs show a high similarity to the manual designs of experienced modelers. This implies that consideration of associated IDEs is an accurate means to approximate design decisions made by human modelers.

5.4. Proposition 3

Proposition 3 states that well-designed activities work towards the production of an IDE (as identified by the five proposed patterns) and consist of a semantically coherent set of operations (as defined in Proposition 2). To evaluate this proposition, the automatically generated activity designs are compared to the solutions. The evaluation of this proposition does not depend on assumptions about the solutions, since it is fair to believe that all activity designs are intended to be *well-designed*. Thus, all solutions are included in the evaluation of Proposition 3.

5.4.1. Metrics

The metrics used in the evaluation of this proposition are equal to those used for the evaluation of Proposition 2.

5.4.2. Results

As shown in Table 6, the scores for the Rand index range from 0.81 to 0.98 between cases. These scores are relatively stable when

Table 4
Evaluation results for Proposition 1.

	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 5
Evaluation results for Proposition 2.

	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
Evaluation results for Proposition 3.

	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

compared to the Jaccard index, which ranges from 0.46 to 0.95. A high Rand index and low Jaccard index on the same solution (e.g. for the BICL case), indicates that true negatives (i.e. correctly identified dissimilarity of operations) have a large impact on the outcome. This happens when the proposed design and the solution are both fine-grained, but the latter consists of even smaller activities.

The variance in the evaluation scores calls for an investigation into the causes of the difference between generated and manual activity designs. We discuss these differences in Section 5.5.

5.5. Discussion of results

The evaluation in the previous sections show that automatically generated designs generally have a high similarity to the manually designed solutions found in literature. In other words, the proposed guidelines approximate the design choices made by experienced modelers.

The evaluation of Proposition 1 reveals that the generated designs are, on average, more coarse-grained than 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, hand-overs of work are not captured in the generated designs. Human modelers, by contrast, incorporate these hand-overs intuitively, based on the data element descriptions. This results in a coarser process granularity for the proposed design, due to the limited amount of explicit information that is used to generate the activities.

Section 5.3 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 because we maintain the visibility of important decision logic in process models. By contrast, some solutions obscure this logic and hide the decisions inside larger activities. It is noteworthy that without these differences, the solutions are nearly identical to the proposed designs. This implies that the notion of associated IDEs closely approximates the way in which human modelers assign group operations into activities once they have determined the desired output element of an activity.

The evaluation of Proposition 3 shows that the automatically generated designs are for some cases valid approximations of the solutions, but the performance is worse for the BICL and UNEM cases. For the BICL case, this is due to differing process granularity. As previously indicated, the solutions for this case, intuitively, included resource requirements, while the generated designs do not capture such implicit information. Another cause of differences between generated designs and the solutions is observed for the UNEM case. Upon inspection, it is revealed that these differences occur due to the large number of automated, i.e. instantaneous and costless, operations in this process (37 out of 51). The solutions are designed such that the automated operations are executed as soon as they are enabled [17]. By contrast, the proposed designs only execute these operations once required. Since the automated operations encompass a majority of the total operations, this distinction has a significant impact on the validation scores. These design differences are irrelevant to the execution efficiency of the process, since the automated operations are instantaneous and costless. However, by treating these operations in the same way as non-instant operations, automated operations are also grouped into semantically coherent activities. We therefore ensure that activities are meaningful and we preserve a high level of process model understandability.

The evaluation demonstrates that the automatically generated designs are good approximations of designs by experienced modelers. Furthermore, the causes for the most important differences can be clearly identified. They occur either due to differing modeling preferences, or the minimal amount of process information that can be objectively incorporated during the generation of the designs.

6. Discussion

The guidelines presented in this work provide the means to objectify and automate the act of activity composition for data-centric workflows. Properly composing activities is important for the design of efficient and understandable workflow processes. However, without clear guidelines and support, this task is dependent on intuition and subjective decisions. For this reason, activity composition builds on expertise and thorough familiarity with the workflow process, and can be altogether a time-consuming act. The contribution of this work, in the form of composition guidelines, lies in the significant reduction of the expertise and time that is required to properly compose a set of activities for a workflow. The results of the quantitative evaluation show that the use of these guidelines provides an objective alternative that approximates the otherwise subjective decisions made by experienced modelers.

6.1. Limitations

The majority of the cases included in the validation are based on existing business processes. Furthermore, they vary greatly in complexity and application domain. Even though this shows that the validation covers a diverse range of real processes, only a limited number of suitable PDMs and accompanying activity designs are found in literature. The strength of the evaluation can thus be increased by incorporating additional cases and activity designs.

The proposed guidelines are purposefully designed to work on a minimal amount of information. As a result, the guidelines exclude information beyond the structure of a PDM, even if such information would be available. The quality of resultant activity designs could be improved by incorporating contextual information (e.g. the organizational perspective of a process) when composing activities. Unfortunately, such information is not present for the available example PDMs. Therefore, aspects beyond the structure of a PDM are excluded from consideration in this work.

As a final limitation, it should be noted that the proposed guidelines are designed and validated in the context of PBWD. Although it is conceivable that the rules can be easily adapted to different representations of data-flow structure, it has not been verified whether such a translation yields the same results as it does for PDMs.

6.2. Future work

The presented research demonstrates the feasibility of objective and automated support for the composition of workflow activities. Several opportunities exist to expand the fundamental guidelines into new directions.

First, the existing guidelines can be extended by incorporating information beyond the structural properties of a PDM. A big

challenge here lies in the inclusion of guidelines that consider the context of a process, e.g. guidelines that set a maximum execution time for activities. A suitable maximum time depends on the availability of a large amount of information related to a process and its context. Due to this, such guidelines are inherently more complex than the context-free guidelines presented in this work.

The composition of workflow activities is the first step required to translate a PDM into a workflow process. The presented work, however, does not asses the next step: converting an activity design into a process model. Future research could focus on an approach that converts automatically generated activity designs into process models. Although generic translation techniques exist [18], a technique could be designed that specifically considers the characteristics of activities designed in accordance with the proposed guidelines. For example, the activities are designed such that they allow efficient process execution through concurrency. By extending the activity composition approach with a step that generates suitable process models, the transformation from data-flow structure to workflow process would become fully supported.

Finally, future work could explore the applicability of the composition guidelines in related research areas. The area of *process mining* appears to be particularly interesting in this respect. Process mining aims to extract information about processes from so-called *event logs*. These logs consist of large quantities of atomic processing steps that occur during process execution. The sheer size and granularity of the data in event logs can distort the understandability of process mining results [26]. The concepts used to compose activities may be able to identify and group related events. Thereby, a higher level and better comprehensible view of the data can be generated. This would render the application of the guidelines in process mining contexts a promising direction.

7. Related work

The presented work focuses on the grouping of elementary data processing steps into proper and meaningful activities. This work extends the state of the art in several ways.

In recent years a number of data-centric process modeling methods have been proposed, e.g. [15,14,13,27–29]. In all of these approaches however, the issue of how to identify and compose activities out of smaller operations and data elements does not play a role. For instance in the approach by Müller et al. [14] (the one closest to the PBWD approach) standard activities (such as develop, test, release) are used and orchestrated into lifecycles for each of the (intermediate) products that are described in the Bill-of-Material. Although these lifecycles (can) have inter-relations, there exist no composition approaches that group together activities related to multiple intermediate products.

In the activity-oriented modeling approaches, the interaction between the steps in a process and the data being processed is at the basis of a wide range of research efforts [11,30,31]. Yet, their exclusive focus is on the detection of data-flow errors. While we agree that it is important to ensure that a workflow functions correctly, they leave open the issue of designing granular, meaningful tasks. By contrast, literature on *job design* does consider the meaningfulness of activities by introducing notions such as *task identity*, *task significance* and *experienced meaningfulness* [5]. However, such notions are rather abstract and do not rest on a detailed understanding of the data-flow perspective of a process. They therefore do not provide objective guidance when grouping elementary processing steps. Job design also plays an important role in research on *assembly line balancing* for manufacturing processes. Assembly line balancing aims to establish an optimal distribution of workload over work stations along an assembly line. These mathematical models depend on detailed insights into, e.g. *operation times*, *factory layout*, *resource availability*, and *utilization* [32]. In case the necessary information is available, its exclusive focus on execution efficiency leaves potential to combine assembly line balancing models with our work that also considers meaningfulness and understandability. For instance, by further refining activities composed according to our guidelines in order to ensure their processing times result in an optimal execution efficiency.

Finally, the presented work relates to research on business process model abstraction (BPMA) and fragmentation. By applying the composition guidelines proposed in this work, a quick overview of the most important parts of the workflow process and their inter-relations is created. This has been found to be a highly demanded use case for BPMA [33]. Other works that address this use case exist. E.g. the work by Polyvyanyy et al. [34], proposes an approach to decompose process models based on structural properties. Their work shares our proposition that structural aspects of a process model can be used to distinguish the main features of a model from insignificant details. Smirnov et al. [35] acknowledge that experienced process modelers take a wider range of properties into account than just a model's structure. Their approach focuses on semantic similarity based on the shared use of documents and resource classes. The former aspect is closely related to data flow, which forms a core concept in our research. Leopold et al. [36] further introduce naming strategies to automatically generate labels for aggregated process elements. Their proposition that meaningful labels improve understandability of abstracted process models is here recognized and hence incorporated in our implementation.

Whereas abstraction generally groups process elements to gain a different view-point or increase the understandability, business process model fragmentation techniques split a process into smaller model fragments with the intention to distribute the fragments over different execution and controlling partners [37]. Work including [e.g. 38–41] define (automated) derivation rules to transform a process into multiple process parts. These parts can then be used to distribute process visibility and execution over multiple engines, resulting in a more flexible environment. Fragmentation uses a similar method (grouping process elements) for a different goal than our composition guidelines. BPMF nevertheless provides an interesting additional perspective to consider when composing workflow activities.

8. Conclusions

The proper composition of activities is important for the design of efficient, meaningful and understandable workflow processes. As to date, this task is time-consuming and requires expertise as well as extensive domain knowledge. To overcome these impediments, this paper introduced fundamental guidelines for the automated composition of activities. The guidelines stipulate that activities should work towards the production of an important piece of information and should consist of semantically related processing steps. The propositions further encompass means to objectively derive these properties based on structural data-flow relations, as captured in a PDM. Due to their exact and precise nature, the guidelines can be used to generate activity designs in a fully automated manner. The usefulness of the guidelines was evaluated by applying them on a real-life business process, as well as through a quantitative validation. The evaluation showed that the proposed guidelines closely approximate design decisions made by experienced modelers. Thereby, the guidelines have been shown to provide an objective and automated alternative to the complex and time-consuming task of the manual composition of activities.

References

[1] W.M.C. Specification, Workflow Management Coalition, Terminology & Glossary (Document No. WFMC-TC-1011), Workflow Management Coalition Specification, 1999.
[2] A. Seidmann, A. Sundararajan, The effects of task and information asymmetry on business process redesign, Int. J. Prod. Econ. 50 (2) (1997) 117–128.
[3] I. Vanderfeesten, H.A. Reijers, W.v.d. Aalst, Evaluating workflow process designs using cohesion and coupling metrics, Comput. Ind. 59 (5) (2008) 420–437.
[4] H.A. Reijers, I. Vanderfeesten, Cohesion and coupling metrics for workflow process design, Bus. Process Manag. (2004) 290–305.
[5] J. Hackman, G. Oldham, Motivation through the design of work: test of a theory, Organ. Behav. Hum. Perform. 16 (2) (1976) 250–279.
[6] Y. Fried, G.R. Ferris, The validity of the job characteristics model: a review and meta-analysis, Pers. Psychol. 40 (2) (1987) 287–322.
[7] J. Mendling, H. Reijers, W.v.d. Aalst, Seven process modeling guidelines (7pmg), Inf. Softw. Technol. 52 (2) (2010) 127–136.
[8] A. Polyvyanyy, S. Smirnov, M. Weske, Business process model abstraction, Handb. Bus. Process Manag. 1 (2010) 149–166.
[9] I. Vanderfeesten, Product-Based Design and Support of Workflow Processes, Eindhoven University of Technology, 2009 (Ph.D. thesis).
[10] H. van der Aa, H.A. Reijers, I. Vanderfeesten, Composing workflow activities on the basis of data-flow structures, in: Business Process Management, Springer, 2013, pp. 275–282.
[11] S.X. Sun, J.L. Zhao, J.F. Nunamaker, O.R.L. Sheng, Formulating the data-flow perspective for business process management, Inf. Syst. Res. 17 (4) (2006) 374–391.
[12] H.A. Reijers, S. Limam, W.v.d. Aalst, Product-based workflow design, J. Manag. Inf. Syst. 20 (1) (2003) 229–262.
[13] H. Reijers, S. Limam Mansar, W.v.d. Aalst, Product-based workflow design, J. Manag. Inf. Syst. 20 (1) (2003) 229–262.
[14] D. Müller, M. Reichert, J. Herbst, Data-driven modeling and coordination of large process structures, in: R. Meersman, Z. Tari (Eds.), Proceedings of the 15th International Conference on Cooperative Information Systems (CoopIS'07), vol. 4803, 2007, pp. 131–149.
[15] A. Nigam, N. Caswell, Business artifacts: an approach to operational specification, IBM Syst. J. 42 (3) (2003) 428–445, http://dx.doi.org/10.1147/sj.423.0428.
[16] J. Orlicky, Structuring the bill of materials for MRP, Prod. Inventory Manag. (1972) 19–42.
[17] H.A. Reijers, Design and Control of Workflow Processes: Business Process Management for the Service Industry, Springer-Verlag, 2003.
[18] I. Vanderfeesten, H.A. Reijers, W.v.d. Aalst, J. Vogelaar, Automatic support for product based workflow design: generation of process models from a product data model, in: OTM 2010 Workshops, Springer, 2010, pp. 665–674.
[19] C. Jarrett, G. Gaffney, Forms that Work: Designing Web Forms for Usability, Morgan Kaufmann Pub., 2008.
[20] H. Van der Aa, Composing Workflow Activities, Eindhoven University of Technology, 2013 (Master's thesis).
[21] H.A. Reijers, S. Liman Mansar, Best practices in business process redesign: an overview and qualitative evaluation of successful redesign heuristics, Omega 33 (4) (2005) 283–306.
[22] H. Diaz Garcia, Evaluation of Data-Centric Process Modeling Approaches, Eindhoven University of Technology, 2011 (Master's thesis).
[23] R. Baeza-Yates, B. Ribeiro-Neto, et al., Modern Information Retrieval, vol. 463, ACM Press, New York, 1999.
[24] L. Hubert, P. Arabie, Comparing partitions, J. Classif. 2 (1) (1985) 193–218.
[25] G.W. Milligan, A Monte Carlo study of thirty internal criterion measures for cluster analysis, Psychometrika 46 (2) (1981) 187–199.
[26] C. Günther, W.v.d. Aalst, Mining Activity Clusters From Low-Level Event Logs, BPM Center Report BPM-06-11, BPMcenter.org, 2006.
[27] J. Wang, A. Kumar, A framework for document-driven workflow systems, in: W.v.d. Aalst, B. Benatallah, F. Casati, F. Curbera (Eds.), Proceedings of the 3rd International Conference on Business Process Management (BPM 2005), Lecture Notes in Computer Science, vol. 3649, (2005), pp. 285–301.
[28] J. Küster, K. Ryndina, H. Gall, Generation of business process models for object life cycle compliance, in: G. Alonso, P. Dadam, M. Rosemann (Eds.), Proceedings of the 5th International Conference on Business Process Management (BPM 2007), vol. 4714, 2007, pp. 165–181.
[29] V. Künzle, M. Reichert, Striving for object-aware process support: how existing approaches fit together, in: 1st Int'l Symposium on Data-driven Process Discovery and Analysis (SIMPDA'11), 2011, http://dbis.eprints.uni-ulm.de/745/.
[30] S. Sadiq, M. Orlowska, W. Sadiq, C. Foulger, Data flow and validation in workflow modelling, in: in: Proceedings of the 15th Australasian database conference, vol. 27, Australian Computer Society, Inc., 2004, pp. 207–214.
[31] N. Trčka, W.v.d. Aalst, N. Sidorova, Data-flow anti-patterns: discovering data-flow errors in workflows, in: in: Advanced Information Systems Engineering, Springer, 2009, pp. 425–439.
[32] C. Becker, A. Scholl, A survey on problems and methods in generalized assembly line balancing, Eur. J. Oper. Res. 168 (3) (2006) 694–715.
[33] S. Smirnov, H. Reijers, T. Nugteren, M. Weske, Business Process Model Abstraction: Theory and Practice, No. 35, Universitätsverlag Potsdam, 2010.
[34] A. Polyvyanyy, S. Smirnov, M. Weske, On application of structural decomposition for process model abstraction, in: in: 2nd Int. Conf. BPSC, Citeseer, 2009, 110–122.
[35] S. Smirnov, H. Reijers, M. Weske, A semantic approach for business process model abstraction, in: in: Advanced Information Systems Engineering, Springer, 2011, pp. 497–511.
[36] H. Leopold, J. Mendling, H. Reijers, On the automatic labeling of process models, in: in: Advanced Information Systems Engineering, Springer, 2011, pp. 512–520.
[37] R. Khalaf, O. Kopp, F. Leymann, Maintaining data dependencies across bpel process fragments, Int. J. Coop. Inf. Syst. 17 (03) (2008) 259–282.
[38] G.B. Chafle, S. Chandra, V. Mann, M.G. Nanda, Decentralized orchestration of composite web services, in: in: Proceedings of the 13th International World Wide Web Conference on Alternate Track Papers & Posters, ACM, (2004), pp. 134–143.
[39] Y. Zhai, H. Su, S. Zhan, A data flow optimization based approach for BPEL processes partition, in: in: IEEE International Conference on e-Business Engineering, 2007. ICEBE 2007, IEEE, (2007), pp. 410–413.
[40] P. Hens, M. Snoeck, G. Poels, M. De Backer, Process fragmentation, distribution and execution using an event-based interaction scheme, J. Syst. Softw. 89 (2014) 170–192.
[41] G. Li, V. Muthusamy, H.-A. Jacobsen, A distributed service-oriented architecture for business process execution, ACM Trans. Web (TWEB) 4 (1) (2010) 2.

Han van der Aa is a PhD candidate in the Business Informatics group at VU University Amsterdam, the Netherlands. He received his M.Sc. degree in Business Information Systems from the Eindhoven University of Technology (TU/e) in 2013. Before joining the VU University Amsterdam, he has been a visiting researcher at the Institute for Information Business at Wirtschaftsuniversität Wien, Austria. His research interests include business process modeling, comparison and transformation of different process representations, natural language processing, and product based workflow design.

Hajo Reijers is a full professor in Business Informatics at VU University Amsterdam, and a part-time full professor in Business Process Technology at Eindhoven University of Technology. He worked in industry as a management consultant for Deloitte and led an R&D group of Lexmark. He published over 150 scientific papers, chapters in edited books, and articles in professional journals on topics such as business process management, workflow technology, process mining, conceptual modeling, and simulation.

Irene Vanderfeesten is an assistant professor in Business Process Management and Information Systems at Eindhoven University of Technology (TU/e). She received her MSc degree in Computer Science and her PhD degree in Industrial Engineering from TU/e in 2004 and 2009, respectively. Before joining the Information Systems group of the department of Industrial Engineering and Innovation Sciences at TU/e in 2010 she also worked as an IT consultant in the banking and insurance sector. Her current research interests include business modeling; business process redesign and improvement; workflow management; and human aspects of business process management and information systems.