Softw Syst Model
DOI 10.1007/s10270-013-0356-2

SPECIAL SECTION PAPER

Investigating expressiveness and understandability of hierarchy in declarative business process models

Stefan Zugal · Pnina Soffer · Cornelia Haisjackl · Jakob Pinggera · Manfred Reichert · Barbara Weber

Received: 30 September 2012 / Revised: 29 March 2013 / Accepted: 27 May 2013

Abstract Hierarchy has widely been recognized as a viable approach to deal with the complexity of conceptual models. For instance, in declarative business process models, hierarchy is realized by sub-processes. While technical implementations of declarative sub-processes exist, their application, semantics, and the resulting impact on understandability are less understood yet—this research gap is addressed in this work. More specifically, we discuss the semantics and the application of hierarchy and show how sub-processes enhance the expressiveness of declarative modeling languages. Then, we turn to the influence of hierarchy on the understandability of declarative process models. In particular, we present a cognitive-psychology-based framework that allows to assess the impact of hierarchy on the understandability of a declarative process model. To empirically test the proposed framework, a combination of quantitative and qualitative research methods is followed. While statistical tests provide numerical evidence, think-aloud protocols give insights into the reasoning processes taking place when reading declarative process models.

Keywords Business process management · Declarative business process models · Modularization · Understandability · Cognitive psychology

Communicated by Dr. Selmin Nurcan.

This research is supported by Austrian Science Fund (FWF): P23699-N23.

S. Zugal (✉) · C. Haisjackl · J. Pinggera · B. Weber
University of Innsbruck, Innsbruck, Austria
e-mail: stefan.zugal@uibk.ac.at

C. Haisjackl
e-mail: cornelia.haisjackl@uibk.ac.at

J. Pinggera
e-mail: jakob.pinggera@uibk.ac.at

B. Weber
e-mail: barbara.weber@uibk.ac.at

P. Soffer
University of Haifa, Haifa, Israel
e-mail: spnina@is.haifa.ac.il

M. Reichert
University of Ulm, Ulm, Germany
e-mail: manfred.reichert@uni-ulm.de

1 Introduction

Using modularization to hierarchically structure information has for decades been identified as a viable approach to deal with complexity [31]. Not surprisingly, business process modeling languages provide support for hierarchical structures, e.g., sub-processes in BPMN [29] and YAWL [55]. However, in general, *"the world does not represent itself to us neatly divided into systems, subsystems. . . these divisions which we make ourselves"* [16]. In this sense, a viable discussion about the proper use of modularization for the analysis and design of information systems as well as its impact on understandability is still going on. In business process management (BPM), sub-processes have been recognized as an important factor influencing model understandability [10]; however, there are no definitive guidelines on their use yet. For instance, recommendations regarding the size of a sub-process in an *imperative process model* range from 5–7 model elements [48] over 5–15 model elements [21] to up to 50 model elements [26]. For declarative process models, which have recently gained attention due to their flexibility [34,44], the proper usage of modularization has not been investigated at all. While work has been done with respect to the technical support of declarative sub-processes, it remains

Published online: 20 June 2013

unclear *whether and when* hierarchy has an influence on the *understandability* of the process model. In general, empirical research into the understandability of conceptual models, such as ER diagrams or UML statecharts, has shown that hierarchy can have a positive influence [45], negative influence [5], or no influence at all [6]. For declarative process models, however, no respective empirical studies have been conducted so far, hence the situation is less clear. However, as declarative process models appear to be especially challenging to understand, it seems particularly important to improve their understandability. For instance, in [33] it is argued that due to the interconnections between constraints, declarative process models quickly can become too complex for humans to deal with. Similarly, [68] points out that *hidden dependencies*, i.e., dependencies between constraints that are not directly visible, may hamper the understanding of declarative process models. In the following, we will shed light on the question which influence on understandability can be expected for hierarchy in declarative process models.

The contribution of this work is twofold. First, the semantics of hierarchy in declarative process models is elaborated on. In particular, we will show that hierarchy is not just a question of structure, but also enhances expressiveness and has implications on the restructuring of a model. Second, the impact of hierarchy on the understandability of the model will be investigated systematically. We will present a cognitive-psychology-based framework that explains general effects of hierarchy, but also takes peculiarities of declarative process models into account. The framework allows to assess the possible impact of hierarchy, i.e., whether a certain modularization of a declarative process model has a positive influence, negative influence, or no influence at all. To test these claims empirically, we follow a combination of quantitative and qualitative research methods.[1]

This paper extends the results of [69] primarily by conducting an empirical evaluation of the proposed framework for assessing understandability. This, in turn, allows to advance the previous work in three dimensions. First, it provides the indispensable empirical validation of the framework. Second, the qualitative nature of the investigation provides valuable insights into the understanding of declarative process models. In that ways, this extension also contributes to the still developing field of research into the declarative process modeling paradigm (cf. [36]). Third, discussions which have been theory-based in [69] are enriched with empirical findings.

The remainder of this paper is structured as follows. Section 2 introduces declarative process models. Then, Sect. 3 discusses the semantics of hierarchy in declarative process models. Subsequently, Sect. 4 deals with the application of hierarchy in declarative process models, whereas Sect. 5 investigates the impact on understandability and Sect. 6 discusses limitations. Finally, related work is presented in Sect. 7 and the paper is concluded with a summary and an outlook in Sect. 8.

[1] Please note that even though we take declarative models in general into account, we will make use of the declarative language ConDec [33] for the discussion.

2 Background: declarative process models

There has been a long tradition of modeling business processes in an imperative way. Process modeling languages supporting this paradigm, like BPMN, EPC, and UML Activity Diagrams, are widely used. Recently, *declarative approaches* have received increasing interest and suggest a fundamentally different way of describing business processes [33]. While imperative models specify exactly *how* things have to be done, declarative approaches only focus on the logic that governs the interplay of actions in the process by describing the *activities* that can be performed, as well as *constraints* prohibiting undesired behavior. An example of a constraint in an aviation process would be that crew duty times cannot exceed a predefined threshold. Constraints described in literature can be classified as execution constraints and completion constraints (also referred to as termination constraints, cf. [69]). *Execution* constraints, on the one hand, restrict the execution of activities, e.g., an activity can be executed at most once. *Completion* constraints, on the other hand, affect the completion of process instances and specify when process completion is possible. For instance, an activity must be executed at least once before the process can be completed. Most constraints focus either on execution *or* completion semantics; however, some constraints also combine execution and completion semantics (e.g., the succession constraint [33]).

To illustrate the concept of declarative processes, a model (PM_M) specified in ConDec [33] is shown in Fig. 1a. It contains activities *A* to *F* as well as constraints *C*1 and *C*2. *C*1 prescribes that A must be executed at least once (i.e., *C*1 restricts the completion of process instances). *C*2 specifies that *E* can only be executed if *C* has been executed at some point in time before (i.e., *C*2 imposes restrictions on the execution of activity *E*). In Fig. 1b, an example of a process instance (PI_M) illustrates the semantics of PM_M. Therein, we make use of *events* to describe relevant changes during process execution, e.g., instantiation of the process instance or the start and completion of activities. After process instantiation (event *e*1), *A, B, C, D* and *F* can be executed. *E*, however, cannot be executed as *C*2 specifies that *C* must have been executed before (cf. gray bar below "*E*"). Furthermore, the process instance cannot be completed as *C*1 is not satisfied, i.e., *A* has not been executed at least once (cf. gray

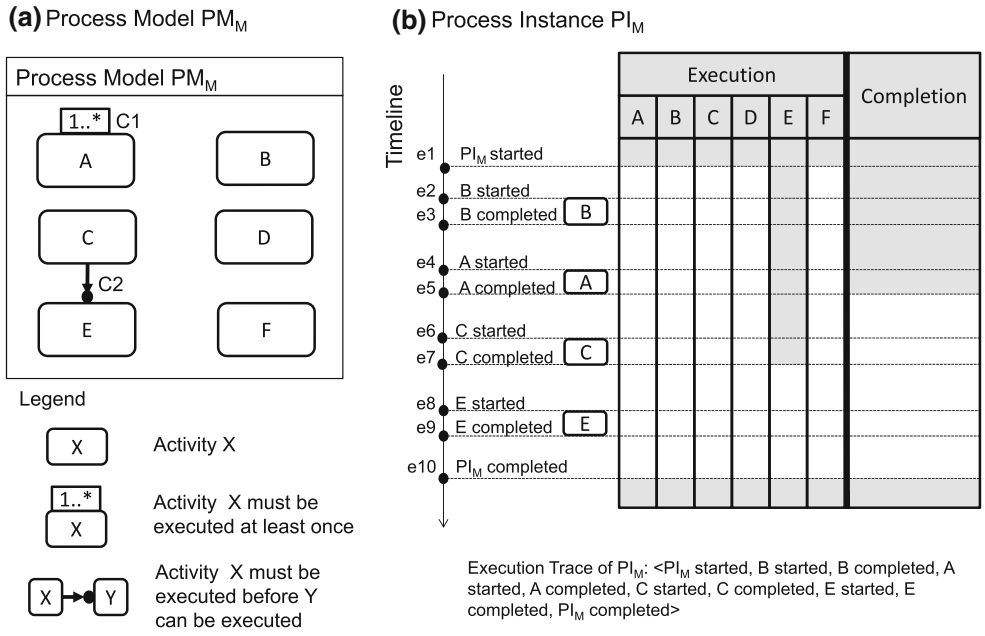


Fig. 1 Executing a declarative process model

area below "Completion"). The subsequent execution of B (in $e2$ B is started, in $e3$ B is completed) does not cause any changes as B is not involved in any constraint. However, after A is executed ($e4$, $e5$), $C1$ is satisfied, i.e., A has been executed at least once and thus PI_M can be completed—after $e5$ the box below "Completion" is white. Then, C is executed ($e6$, $e7$), satisfying $C2$ and consequently allowing E to be executed. Finally, the execution of E ($e8$, $e9$) does not affect any constraint; thus, no changes with respect to constraint satisfaction can be observed. As all completion constraints are satisfied, PI_M can be completed. Please note that declarative process instances have to be completed explicitly, i.e., the end-user must decide when to complete the process instance ($e10$). Completion constraints thereby specify when completion is allowed, i.e., PI_M could have been completed at any point in time after $e5$. As illustrated in Fig. 1b, a process instance can be specified through a list of *events*. In the following, we will denote this list as *execution trace*, e.g., for $\text{PI}_M : \langle e1, e2, e3, \ldots, e10 \rangle$.

3 Background: semantics of sub-processes

This section aims at establishing an understanding of the semantics of sub-processes in a declarative model. In general, a sub-process is introduced in a process model via a *complex activity*, which refers to a process model. When the complex activity is executed, the referred process model, i.e., the sub-process, is instantiated. Thereby, sub-processes are viewed as *separate* process instances, i.e., when a complex activity is started, a new instance of the sub-process the complex activity is referring to, is created (cf. [29,35]). The parent process, however, has no information about the internals of the sub-process, i.e., the sub-process is executed in isolation. In this sense, according to [11,41,42], we view sub-processes from an *integrated perspective*, i.e., the sub-process is seen as a *black box*. Interaction with the parent process is only done via the sub-process' life cycle[2]. Thereby, the life cycle state of the complex activity reflects the state of the sub-process [35], e.g., when the sub-process is in state *completed*, also the complex activity must be in state *completed*.

Considering this, it is essential that sub-processes are executed in isolation, as isolation forbids that constraints can be specified between activities included in different sub-processes. In other words, in a hierarchical declarative process model with several layers of hierarchy, the constraints of a process model *can neither directly* influence the control flow of any parent process, *nor directly influence* the control flow of any (sub-) process on the same layer or a layer below. Please note that control flow may still be *indirectly influenced* by restricting the execution of a sub-process, thereby restricting the execution of the activities contained therein.

To illustrate these concepts, consider the hierarchical process model PM_M in Fig. 2a. It consists of activity A, which has to be executed at least once (cf. constraint $C1$) and complex activity B. B, in turn, refers to process model PM_B, which contains activities C and D. C and D are connected by precedence constraint $C2$, i.e., D can only be executed if C was

[2] We do not take into account the exchange of input- and output data here, as we focus on control flow behavior only.

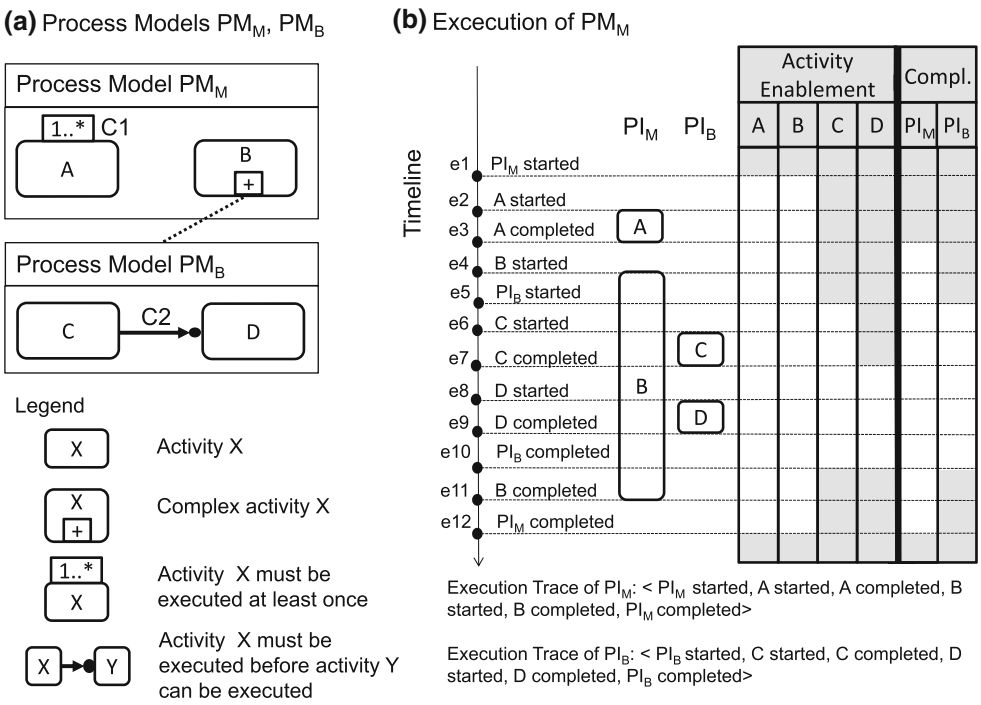


Fig. 2 Execution of a hierarchical declarative process model

executed before. Figure 2b shows an example of an execution of PM_M. On the left, a timeline lists all events that occur during process execution. To the right, the enablement of the activities and whether a process instance can be completed, is illustrated. Whenever the area below an activity / process instance is colored white, it indicates that this activity is currently enabled / the process instance can be completed. The timeline is to be interpreted the following way: By instantiating PM_M ($e1$), activities A and B become enabled, as no constraints restrict their execution. C and D cannot be executed, as they are confined in PM_B and no instance of PM_B is running yet. The subsequent execution of A ($e2$, $e3$) does not change activity enablement, but satisfies the selection constraint on A, hence allowing PI_M to complete. Then, the start of B ($e4$) causes the instantiation of PM_B (PI_B, $e5$). Hence, C becomes enabled, as it can be executed within PI_B. Still, D is not enabled yet as constraint $C2$ is not satisfied. After C is executed ($e6$, $e7$), the precedence constraint is satisfied, therefore also D becomes enabled. After the execution of D ($e8$, $e9$), the user decides to complete PI_B ($e10$), causing C and D to be not executable anymore and triggering the completion of B ($e11$). Still, A and B are enabled as they can be executed within process instance PI_M. Finally, after PI_M is completed by the end user through explicit completion ($e12$), no activity is enabled anymore.

4 Using hierarchy in declarative process models

Regardless of the modeling language, hierarchy allows to structure models and to hide model elements in sub-models. In this section, the use of hierarchy, given the semantics of Sect. 3, is discussed. To illustrate and discuss the implications of hierarchy on declarative process models, we make use of a running example. We chose the business process of writing a scientific paper and created two business process *models* describing the process. In Fig. 3, the process is modeled without hierarchy, whereas in Fig. 4 hierarchical structures are used. We would like to note at this point these models have been created for demonstration purpose and hence might not be perfectly accurate with respect to the modeled domain.

4.1 Preconditions for using sub-processes

While for imperative models any Single-Entry-Single-Exit fragment can be extracted to a sub-process [59,60], in declarative models the structure is not informative enough. Rather, two main conditions should hold for the introduction of sub-processes. First, the activities in a sub-process should relate to a certain intention [52] to be fulfilled. For instance, in Fig. 4, *Read reviews for revising paper*, *Write response letter* and *Work on revision* all serve the purpose of revising a paper. Once the sub-process of *Revise paper* is completed, it is clear that the paper has been revised. On a higher abstraction level it may not make a difference, e.g., how many times *Work on revision* has been executed or whether the reviews have been read. But knowing the paper has been revised is substantial for the continuation of the process. This information is not available in the flat model (and it only exists in the mind of the human who executes the process). Second, the activities included in a sub-process should be such that they can be

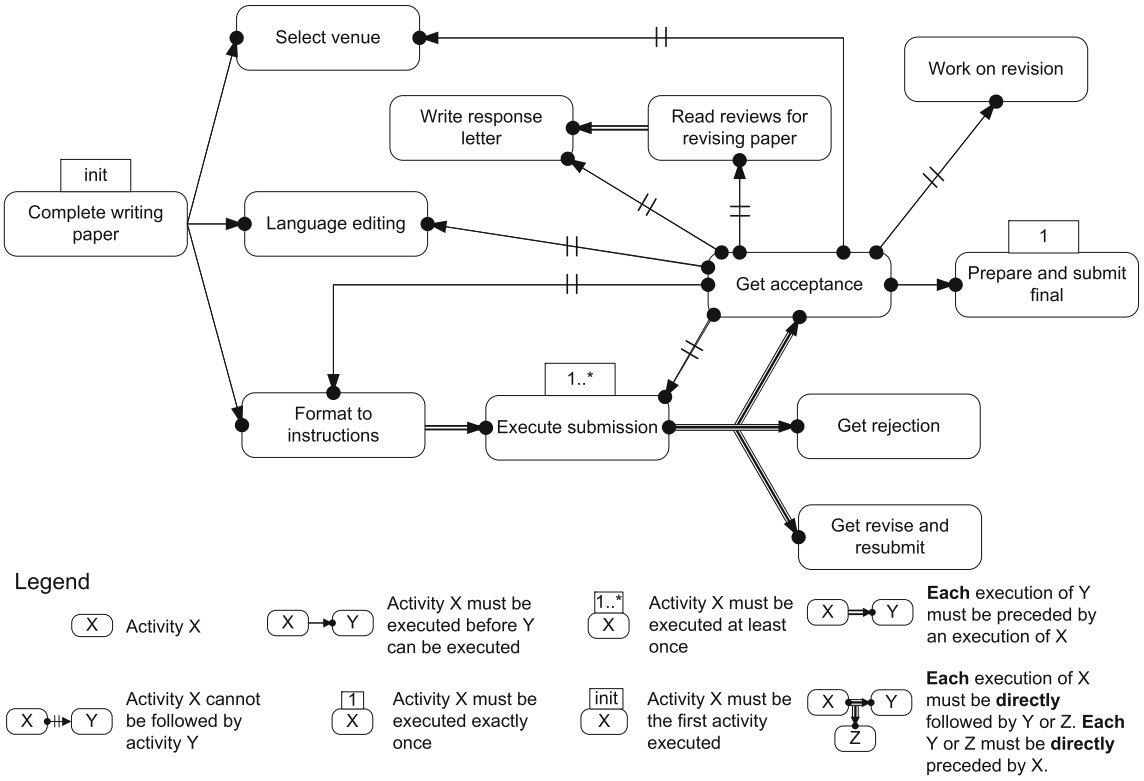


Fig. 3 Example of a flat declarative process model

executed in isolation from the top-level process. This is due to the local nature of the constraints within the sub-process, as discussed in Sect. 3. In other words, a sub-process cannot include any activity that has constraints specifically relating that activity to activities outside the sub-process. Still, if all the activities considered for inclusion in a sub-process share a common constraint with some other activity, then this constraint holds for the entire sub-process. In the flat model (cf. Fig. 3), activities *Read reviews for revising paper*, *Write response letter* and *Work on revision* all have a constraint restricting them from following *Get acceptance*. In the hierarchical model (cf. Fig. 4), these constraints are *aggregated* to one constraint related to the top-level complex activity of *Revise paper*. As the constraints are aggregated to a single constraint, we refer this to as *aggregation of constraints*.

4.2 Enhanced expressiveness

For *imperative* process models, hierarchical decomposition is viewed as a structural measure that may impact model understandability [63], but does not influence semantics. In declarative process models, however, hierarchy also has implications on semantics. More precisely, hierarchy enhances the expressiveness of a declarative modeling language. The key observation is that by specifying constraints that refer to complex activities, it is possible to restrict the life cycle of a sub-process. A constraint that refers to a complex activity thereby not only influences the complex activity, but also all activities contained therein.

This, in turn leads to two effects. First, constraints can be specified that apply for a set of activities (cf. aggregation of constraints in Sect. 4.1). Second, the specification of constraints, that apply in a certain context only, is supported. Consider for instance *Work on revision* and *Revise paper* in Fig. 4. *Work on revision* is mandatory within the context of *Revise paper*. Hence, *Work on revision* must be executed at least once whenever *Revise paper* is executed, but it might not be executed at all (if *Revise paper* is not executed).

To illustrate how these two effects enhance expressiveness, consider models PM_M and PM_C in Fig. 5, which solely use constraints defined in [33]. The chained precedence constraint between C and D specifies that for each execution of D, C and therefore PM_C has to be executed directly before. When executing PM_C, in turn, A has to be executed exactly once and B has to be executed exactly twice (in any order). Hence, the constraint between C and D actually refers to a *set of activities*, i.e., A and B. For each execution of D, A has to be executed exactly once and B has to be executed exactly twice. In other words, constraints on A and B are only valid in the *context of* PM_C. Such behavior cannot be modeled without hierarchy, using the same set of constraints.

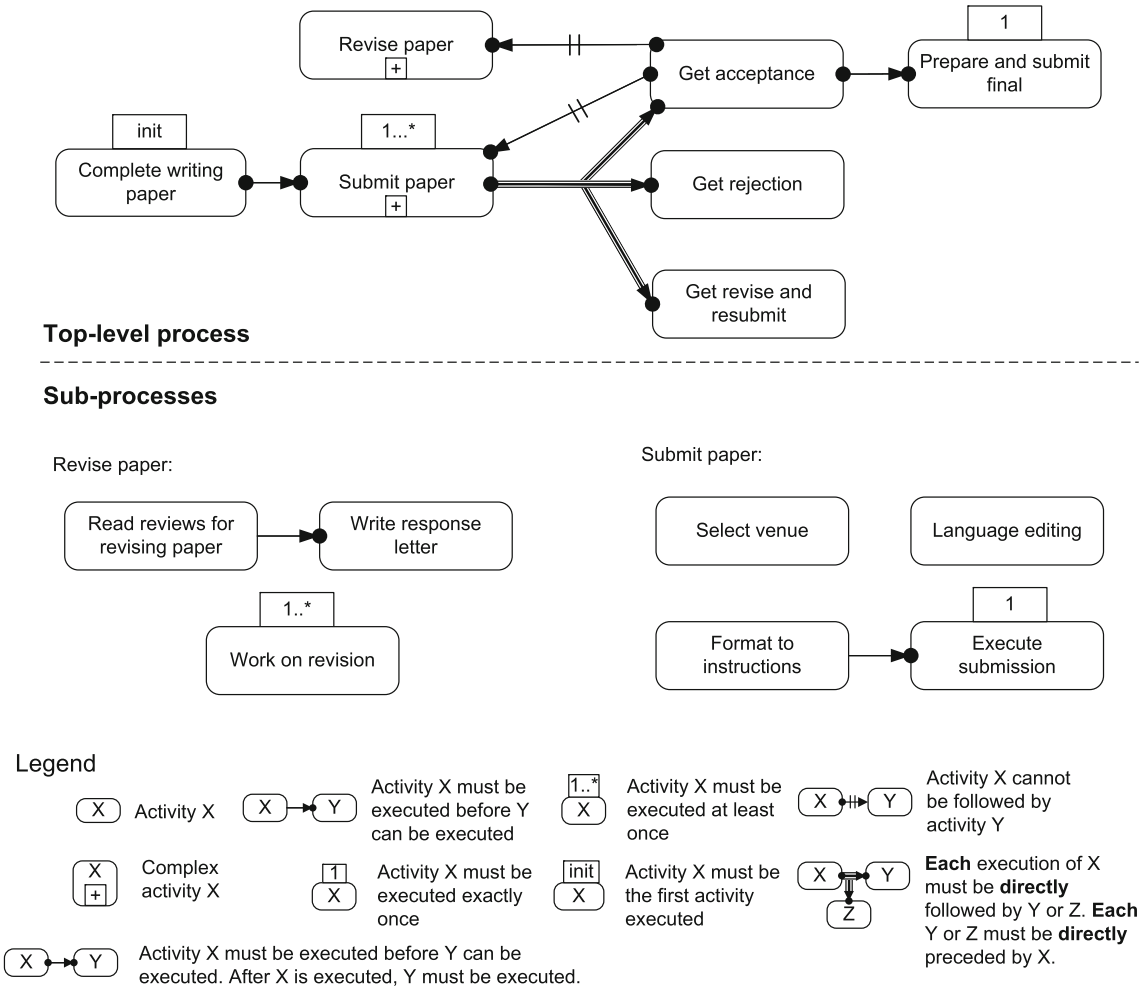


Fig. 4 Example of a hierarchical declarative process model

4.3 Impact on adaptation

Constructing hierarchical models supports top-down analysis, i.e., creating the top-level model first and further refining complex activities thereafter. While this seems like a natural way of dealing with complexity, in some cases, it is desirable to transform a flat model to a hierarchical one. In the following, we will argue why refactoring [59], i.e., changing hierarchical structures in a control-flow preserving way, is only possible under certain conditions for declarative process models. Refactoring requires that *any* hierarchical model can be translated into a model without hierarchy, but the same control-flow behavior (and vice versa). As discussed, expressiveness is enhanced by hierarchy. In other words, there exists control-flow behavior that can be expressed in an hierarchical model, but not in a model without hierarchy—cf. Fig. 5 for an example. Hence, hierarchical models that make use of the enhanced expressiveness cannot be expressed as a flat model, i.e., cannot be refactored.

5 Model understandability

So far, we discussed that hierarchy in declarative process models is not just a question of structure, but also affects semantics. In the following, we will describe how these effects impact the understandability of a declarative process model. In particular, a framework for assessing the impact of hierarchy on understandability is proposed in Sect. 5.1 and empirically tested in Sect. 5.2. Findings and implications are then discussed in Sect. 5.3.

5.1 Framework for assessing understandability

The influence of hierarchy on model understandability has been investigated in a number of different modeling languages, e.g., in ER diagrams [28], imperative business process models [45], and UML statecharts [6] (for an overview see [63]). While reported results do not entirely clarify when and how understandability is affected, a trade-

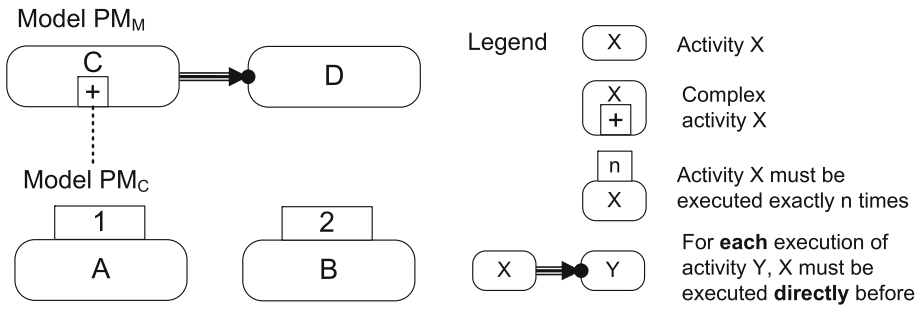


Fig. 5 Example of enhanced expressiveness

off between (sub)model size and degree of hierarchy can be observed. For instance, in small models hierarchy may have no [6] or even a negative impact [5], while for large models a positive influence could be observed [45].

In [63], we introduced a cognitive-psychology-based theory describing *when and why* hierarchy has an impact on understandability (for a introduction to cognitive psychology in business process modeling we refer to [65]). In this work, we present an enhanced version that is still generic but also takes the idiosyncrasies of hierarchy in declarative process models into account. The central concept of the framework is *mental effort* [53], i.e., the mental resources required to solve a problem. In the context of this work, solving a problem refers to understanding the semantics of a declarative process model, i.e., answering questions about a model. According to the framework, hierarchy is the source of two opposing forces influencing this problem solving process. Positively, *abstraction* decreases mental effort by *hiding information* and supporting the *recognition of patterns*. Negatively, *fragmentation* increases mental effort by forcing the analyst to *switch attention* between fragments and *integrating information* from fragments.

5.1.1 Abstraction

Hierarchy allows to aggregate model information by hiding the internals of a sub-process using a complex activity. Thereby, irrelevant information can be hidden from the analyst, leading to decreased mental effort, as argued in [28]. From the perspective of cognitive psychology, this phenomenon can be explained by the concept of *attention management* [23]. During the problem solving process, i.e., answering a question about a model, attention needs to be guided to certain parts of a model. For instance, when checking whether a certain execution trace is supported by a process model, activities that are not contained in the trace are irrelevant for answering the question. Here, abstraction allows removing this irrelevant information, in turn supporting the attention management system and thus reducing mental effort. To illustrate this effect for declarative process models, consider the process model shown in Fig. 4. For answering the question, whether *Get acceptance* can be executed after *Complete writing paper*, it is sufficient to look at activities *Complete writing paper*, *Submit paper*, and *Get acceptance*. In particular, the constraints between those three activities have to be considered, while the content of *Submit paper* is not of interest for this question. In other words, hierarchy helps to abstract from all activities contained in *Submit paper*, making the question easier to answer.

Besides reducing mental effort by improving attention management, abstraction presumably supports the identification of higher level patterns. It is known that the human's perceptual system requires little mental effort for recognizing certain patterns [23,47], e.g., recognizing a well-known person does not require thinking, rather this information can be directly *perceived*. Similarly, in process models, by abstracting and thereby aggregating information, presumably information can be easier perceived. Consider, for example, the process models depicted in Figs. 3, 4. The models are (almost) information equivalent, still we argue that for the model with sub-processes the overall structure and intention of the process is easier to grasp. By introducing complex activities, it is easier to see that the process is about iteratively reworking a paper until it gets accepted. For the sibling-model in Fig. 3, however, the analyst first has to mentally group together activities before the overall intention of the process becomes clear.

5.1.2 Fragmentation

Empirical evidence shows that the influence of hierarchy can range from positive over neutral to negative (cf. [5,6,28,45]). To explain the negative influence, we refer to the *fragmentation* of the model. When extracting a sub-process, modeling elements are removed from the parent model and placed within the sub-process. When answering a question that also refers to the content of a sub-process, the analyst has to *switch attention* between the parent model and the sub-process. In addition, the analyst has to *mentally integrate* the sub-process into the parent model, i.e., interpret constraints in the context of the parent process. From the perspective of cognitive psychology, these phenomena are known to increase mental effort and are referred to as *split-attention effect* [54]. To exemplify this effect, consider the process model in Fig. 4.

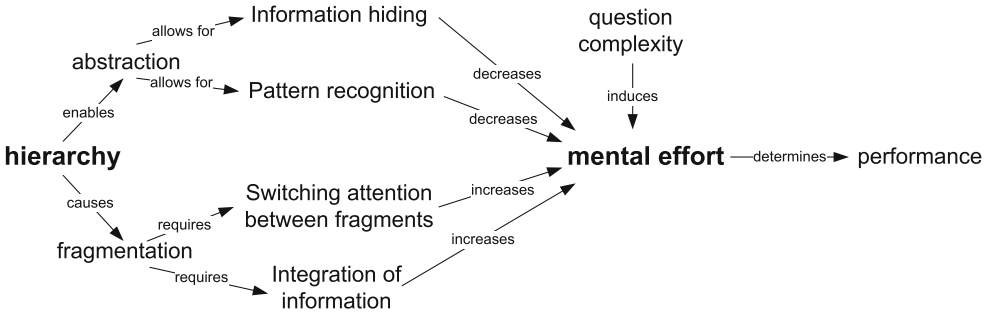


Fig. 6 Framework for assessing hierarchy, adapted from [63]

To determine how often activity *Execute submission* must be executed, it is required to look at activity *Submit paper* too, as *Execute submission* is contained therein. In other words, the analyst has to split attention between these two activities. In addition, the analyst has to integrate the execution semantics of *Submit paper* with the execution semantics of *Execute submission*. Both activities are mandatory, i.e., must be executed at least once, hence for any execution of the overall process, *Execute submission* must be executed at least once. In other words, it is necessary to mentally integrate the constraints restricting the execution of *Submit paper* as well as constraints restricting the execution of *Execute submission*.

Please note that fragmentation is inevitable as soon as modularization is introduced—even for well-modularized models. Consider, for instance, an analyst who wants to find all activities that are assigned to a specific role. In this case, it is very likely that the analyst will have to look through several sub-processes to locate all these activities. Hence, the impact of modularization on the understanding of a model will depend on whether fragmentation can be compensated by abstraction, as detailed in the following.

5.1.3 Interplay of abstraction and fragmentation

According to the model illustrated in Fig. 6, a question's complexity induces a certain mental effort, e.g., locating an activity is easier than validating an execution trace. In addition, mental effort may be decreased by information hiding and pattern recognition or increased by the need to switch between sub-processes and integrate information. Thereby, abstraction as well as fragmentation occur at the same time. A model without sub-processes apparently cannot benefit from abstraction, neither is it impacted by fragmentation. By introducing hierarchy, i.e., creating sub-processes, *both* abstraction and fragmentation are stimulated. Whether the introduction of a new sub-process influences understandability positively or negatively then depends on whether the influence of abstraction or fragmentation predominates. For instance, when introducing hierarchy in a small process model, not too much influence of abstraction can be expected, as the model is small anyway. However, fragmentation will appear, regardless of model size. In other words, hierarchy will most likely show a negative influence or at best no influence for small models (cf. [5,7]).

5.2 Empirical evaluation

Up to now, our framework for assessing the impact of hierarchy on understandability of declarative process models is based on insights from literature. In the following, we will test these claims empirically.

5.2.1 Research questions

The research questions followed in this empirical investigation are derived from the framework presented in Sect. 5.1. In particular, research question 1 (RQ 1) investigates whether analysts are able to understand the semantics of sub-processes. As this requires the analyst to combine the semantics of multiple constraints, it is not obvious a-priori whether such a task is feasible for an average analyst:

> **RQ 1** Do analysts understand the semantics of sub-processes?

Then, research questions 2.1 and 2.2 (RQ 2.1 and RQ 2.2) investigate whether empirical evidence for the positive influence of hierarchy, as postulated in Sect. 5.1, can be found. RQ 2.1 thereby examines the role of pattern recognition, whereas information hiding is approached in RQ 2.2:

> **RQ 2.1** Does pattern recognition support analysts in understanding process models?
> **RQ 2.2** Does information hiding support analysts in understanding process models?

Finally, research question 3 (RQ 3) explores postulated negative effects of hierarchy. In particular, RQ 3 investigates whether fragmentation, i.e., splitting attention and integration of sub-processes, decreases understandability:

> **RQ 3** Does fragmentation hinder analysts in understanding process models?

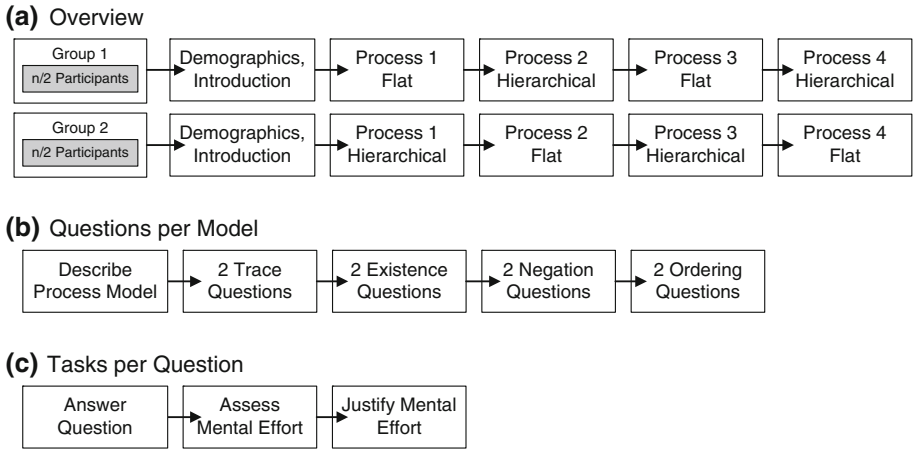


Fig. 7 Experimental design

5.2.2 Experimental design

In order to investigate RQ 1 to RQ 3, we adopt a combination of qualitative and quantitative research methods, as detailed in the following.[3]

Experimental process The experiment's overall process is lined out in Fig. 7a: First, subjects are *randomly*, but evenly, assigned to *Group 1* or *Group 2*. Then, regardless of the group assignment, demographical data are collected and subjects are presented with introductory assignments. To support subjects in their task, sheets briefly summarizing the constraint's semantics are distributed. Data gathered during the introduction are *not* used for analysis. Rather, the introductory tasks allow subjects to familiarize themselves with the type of tasks to be performed—ambiguities can be resolved at this stage without influencing the actual data collection.

After this familiarization phase, subjects are confronted with the actual models designed for data collection. As shown in Fig. 7a, four declarative business processes are used; each of them once modeled with the use of sub-processes and once modeled without sub-processes (the processes are described in detail in paragraph *Experimental Material*). Those four pairs of process models are then distributed between *Group 1* and *Group 2* such that subjects are confronted with hierarchical models and flat models in an alternating manner, cf. Fig. 7a.

As detailed in Fig. 7b, for each model, the same procedure is used. First, the subject is asked to describe what the process is intended to achieve. Second, the subject is confronted with four pairs of questions which have been designed to representatively cover modeling constructs of a declarative process modeling language (details are presented in paragraph *Experimental Material*). For each of the questions, in turn, a three-step procedure is followed, cf. Fig. 7c. First, the subject is asked to answer the question either by *Yes, No* or *Don't Know*. Second, the subject is asked to assess the expended mental effort. To this end, a 7-point rating scale is used, which is known to reliably measure mental effort [17,30]. Third, the subject is asked to explain why it indicated a certain mental effort. Throughout the experiment, subjects are asked to constantly voice their thoughts, i.e., to think-aloud, allowing for a detailed analysis of their reasoning processes [13].

Factor and factor levels Our experiment employs a *two-factorial* design with factor *hierarchy* (factor levels *hierarchical* and *flat*) and factor *impact* (factor levels *abstraction* and *fragmentation*). The elaboration of process models with/without sub-processes realizes factor *hierarchy*, questions formulated according to the framework from Sect. 5.1 realize factor *impact*, as detailed in paragraph *Experimental Material*.

Experimental material The business processes used in this experimental design originate from a case study [18], i.e., describe real-world business processes. From a set of 24 process models collected in the case study, four process models were chosen. In order to make the models amenable for this study, they underwent the following steps. First, the models were translated to English (the case study was conducted in German). Second, inevitable errors occurring in modeling sessions were corrected. Third, the process models had been created without the support of sub-processes. Hence, a second variant of each process was created that describes the same process, but makes use of sub-processes. In Sect. 4.2, we discussed that hierarchy enhances expressiveness in declarative models. In this study design, we refrain from using enhanced expressiveness to keep models comparable.

As summarized in Table 1, process models were chosen such that the number of activities and number of constraints vary. In particular, Process 1 and Process 2 have, compared to

[3] The experimental material can be downloaded from: http://bpm.q-e.at/experiment/HierarchyDeclarative.

Table 1 Characteristics of process models

	Type	Proc. 1	Proc. 2	Proc. 3	Proc. 4
Activities	Flat	11	8	23	23
	Hierarchy	13	9	26	26
Constraints	Flat	19	7	30	45
	Hierarchy	21	9	28	44
Constr. types		8	4	7	5
Sub-processes	Hierarchy	2	1	3	2
Nesting level	Hierarchy	1	1	1	1
Domain		Software dev.	Teaching	Electronic company	Buying an apartment

Process 3 and Process 4, a small number of activities. In addition, all processes have a different number of constraints. The number of activities varies between the flat and hierarchical model, as complex activities had to be introduced in the hierarchical models. Similarly, the number of constraint varies, as processes had to be modeled slightly differently. Since this is the first study investigating sub-processes in declarative models, we decided to keep the model's complexity rather low. In particular, we ensured that not too many different types of constraints (at most 8) and sub-processes (at most 3) were used. Likewise, we decided for a maximum nesting level of 1, i.e., none of the sub-processes referred to another sub-process.

The experiment's questions are designed, as follows. First, for each model, the subject is asked to describe the process model. The idea of this step is to make the subject familiar with the process model to minimize learning effects in the upcoming questions. In addition, by letting subjects freely describe a process model, we intend to get further insights how well models are understood. Second, for each model, 4 categories of representative questions have been designed. In particular, the questions are based on available constraint types [33], i.e., existence, negation, and ordering. In addition, trace questions, i.e., whether an execution trace is valid, are asked to combine aspects of different constraints. For each category of questions, a pair of questions is designed according to the understandability framework from Sect. 5.1. The first question is designed to profit from abstraction, but not being impaired by fragmentation. Hence, the question should be easier to be answered in the hierarchical model than in the flat model. The second question, in turn, is designed to not being profiting from abstraction, but being impaired by fragmentation. Hence, the question should be easier to be answered in the flat model. All in all, for each model, 9 questions are provided—the first one looking into the general understanding of declarative processes, the remaining 8 questions alternatively operationalizing positive and negative effects of hierarchy. Finally, it is ensured that the information provided in the process models is sufficient to answer all questions. In other words, no background knowledge is required for answering questions, as recommended in [32].

Objects The basic *objects* of this experimental design are four declarative business process models, taken from a previous case study on declarative business process modeling [18]. As indicated, the models were pre-processed, to be available in a version with sub-processes and a version without sub-processes, resulting in eight models.

Subjects In order to ensure that measured differences are caused by the impact of hierarchy rather than by unfamiliarity with declarative process modeling, subjects need to be sufficiently trained. Even though we do not require experts, subjects should have a good understanding of declarative processes' principles.

Instrumentation For each question, subjects received separate sheets of paper showing the process model, allowing them to use a pencil for highlighting or taking notes. In addition to recording audio, video recording is used, as video has been proven useful to resolve unclear situations in think-aloud protocols (cf. [62]). Hence, besides collecting quantitative data in terms of answering questions by ternary choices (*Yes*, *No*, *Don't Know*) and measuring mental effort on a 7 point rating scale, qualitative data in terms of think-aloud protocols are gathered.

Response variables The primary response variable of this experimental design is the level of understanding that subjects display with respect to the process models. For operationalization, we measure the mental effort expended for answering questions as well as the amount of correct answers. In addition, think-aloud protocols can be used to analyze errors and their underlying causes in detail.

5.2.3 Experimental execution

Experimental preparation Preparation for the experiment included the elaboration of process models, associated ques-

tions, and the demographical survey. In addition, we prepared material introducing subjects with the tasks to be performed. In case subjects required clarification of a constraint's semantics, we prepared sheets briefly summarizing the semantics of all involved constraints. Finally, models and questions were printed, audio devices and video camera were checked for operability. In parallel, subjects were acquired, and if necessary, trained in declarative process modeling.

Experimental Execution The experiment was conducted in July 2012 in two locations. First, seven subjects participated at the University of Ulm, followed by two additional sessions at the University of Innsbruck, i.e., a total of nine subjects participated. To ensure that subjects were sufficiently familiar with declarative process modeling, all subjects were provided with training material that had to be studied. Each session was organized as follows: In the beginning, the subject was welcomed to the experiment and instructed to speak thoughts out aloud. Since the experimental material consisted over 100 sheets of paper containing process models and questions, we needed to ensure that subjects were not distracted by the extent of material to be processed. To this end, one supervisor was seated left to the subject, a second supervisor to the right, and the sheets containing the experimental material were then passed from the left to the subject. As soon as the subject had finished the task, it passed the sheets further to the supervisor to the right. Hence, no more than a handful of sheets were presented to subjects at once. Meanwhile, a video camera video-recorded the subject's activities and audio-recorded any uttered thoughts. At the end of each session, a discussion followed in order to help subjects reflect on the experiment and to provide us with feedback.

Data Validation In each session, only a single subject participated; hence, we could easily ensure that the experimental setup was obeyed. In addition, we screened whether subjects fitted the targeted profile, i.e., were familiar with BPM and ConDec [33] in particular; results are summarized in Table 2. Demographical questions 1–4 dealt with general knowledge about BPM, i.e., years of modeling experience (avg. 4.9), the amount of models read in the last year (avg. 75.6), the amount of models created last year (avg. 24.0), and the average amount of activities those models contained (avg. 17.7). It can be said that the participants had a profound background in BPM; in fact, the least experienced subject had 2.5 years of modeling experience. Questions 5–7 were concerned with ConDec in particular and were rated on a 7-point Likert Scale, ranging from *"Strongly agree"* (7) over *"Neutral"* (4) to *"Strongly disagree"* (1). Subjects were averagely familiar with ConDec (avg. 3.8), averagely confident in understanding ConDec (avg. 4.1), and averagely confident in creating ConDec models (avg. 4.1). Questions 8–11 assessed the domain knowledge of subjects, as it is known that it can have a significant influence on performance [20]; the same 7-point Likert scale as for question 5–7 was used. Familiarity with software development (Process 1) was on average 5.8, familiarity with teaching (Process 2) on average 5.6, familiarity with electronic companies (Process 3) 3.0, and familiarity with buying apartments (Process 4) 3.6. Finally, we assessed the subjects' professional background: All subjects indicated an academic background.

Table 2 Demographics

	Minimum	Maximum	Mean
1. Years of modeling experience	2.5	7.0	4.9
2. Models read last year	10.0	250.0	75.6
3. Models created last year	5.0	100.0	24.0
4. Average activities	5.0	50.0	17.7
5. Familiarity ConDec	2.0	6.0	3.8
6. Confidence understanding ConDec	2.0	6.0	4.1
7. Confidence creating ConDec	2.0	6.0	4.1
8. Familiarity software development	4.0	7.0	5.8
9. Familiarity teaching	4.0	7.0	5.6
10. Familiarity electronic companies	1.0	6.0	3.0
11. Familiarity buying apartments	1.0	6.0	3.6

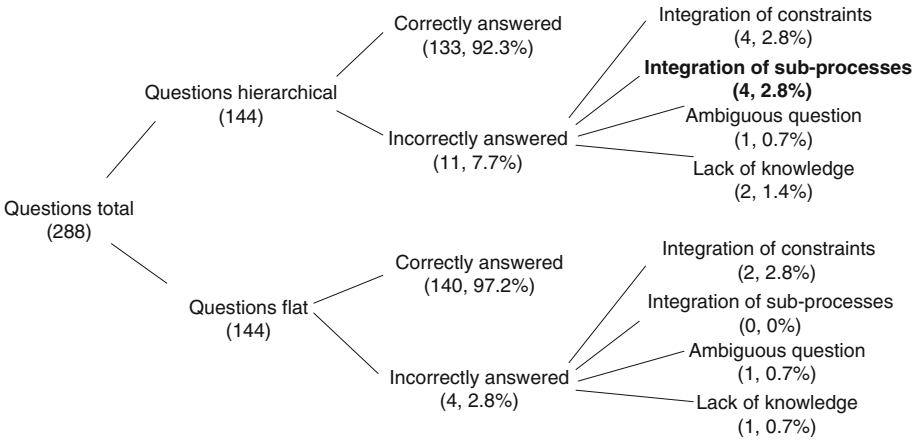
Up to now, we have discussed the design and execution of the empirical study and looked into the demographical data. In the following, we use the gathered data to investigate RQ 1 to RQ 3.

5.2.4 RQ 1: Do analysts understand the semantics of sub-processes?

As discussed in Sect. 5.1, using hierarchy means to *abstract* certain parts of a declarative process model by the means of sub-processes. However, as soon as the content of a sub-process is of concern, the sub-process has to be *integrated* back into the parent process. For a declarative process model, this implies that the semantics of constraints referring to the sub-process and constraints *within* the sub-process have to be combined. As argued, this task might not be trivial; hence, in RQ 1, we investigate whether analysts are basically able to perform this integration task.

In the following, we approach RQ 1 in two steps. First, we classify questions with respect to correctness, i.e., whether a question was answered correctly. Then, we turn toward the think-aloud protocols to investigate error sources. As illustrated in Fig. 8, in total, 288 questions were asked in this experiment (9 subjects × 4 models × 8 questions = 288). In the following, we inspect the upper branch in which questions asked for hierarchical models are summarized. In total, 144 questions were asked for hierarchical models, of which 133 (92.3 %) were answered correctly and 11 (7.7 %)

Fig. 8 Distribution of errors



were answered incorrectly. Apparently, less questions were answered incorrectly in flat models: 4 out of 144 (2.8 %). However, when looking into error sources, it becomes clear that hierarchy is responsible only for a fraction of incorrect answers. In particular, 4 (2.8 %) errors could be traced back to integration of constraints, i.e., when subjects had to combine the semantics of several constraints in order to answer a question. Another 1 (0.7 %) question was answered incorrectly due to an ambiguous wording, i.e., the subject misunderstood the wording of a question. Two (1.4 %) questions were answered incorrectly due to insufficient knowledge about declarative process models. Finally, 4 (2.8 %) questions could be traced back to the presence of hierarchy, i.e., were answered incorrectly because subjects did not properly understand the meaning of constraints in sub-processes in the context of the parent process. In other words, in these cases, subjects had troubles understanding the semantics of the sub-process.

The main findings are hence as follows: First, analysts averagely familiar with ConDec (cf. Table 2) are reasonably capable of interpreting ConDec models, as indicated by the fact that 273 out of 288 (94.8 %) questions were answered correctly. Second, the collected data indicate that analysts are capable of interpreting hierarchical models (133 out of 144 question correct, 92.3 %), only 4 questions (2.8 %) were answered incorrectly *due to* hierarchy. Therefore, we conclude that averagely trained analysts are able to interpret hierarchical declarative process models—however, hierarchy might also be a potential error source. This finding is also inline with the framework presented in Sect. 5.1, i.e., hierarchy is feasible, but has to be applied carefully.

Besides showing that hierarchy is feasible, these findings are also relevant for declarative process models in general. In particular, it has been claimed that process models with a large number of constraints are hard to understand, as the analyst has to keep track of all constraints [33,68]. When analyzing the distribution of errors in Fig. 8, this assumption is further substantiated. In particular, without considering errors conducted due to hierarchy, 11 errors were committed in total. Thereof, 5 errors can be attributed to problems with the experimental execution, i.e., in 2 cases a question was worded ambiguously and in further 3 cases the subject was hindered by lacking knowledge about declarative process models. The remaining 6 errors were classified as *"integration of constraints"*, i.e., when subjects had to integrate the semantics of several constraints. Hence, it can be concluded that problems in understanding are not caused by single constraints, rather the interplay of several constraints seems to pose a significant challenge. Given this finding, it seems plausible that the computer-based *automated* interpretation of constraints can lead to significant improvements in the maintenance of declarative process models [64,67] and the execution of declarative process models [61]. Having established that analysts are able to understand the semantics of sub-processes, we now turn to the question in how far the adoption of hierarchy generates positive effects.

5.2.5 RQ 2.1: Does pattern recognition support analysts in understanding process models?

In Sect. 5.1, we argued that hierarchy supports the analyst in understanding the overall intention of a process. In the following, we will approach this research question in two steps. First, we use think-aloud protocols to identify patterns in understanding declarative process models. Then, we analyze in how far sub-processes support this process of understanding and how it relates to the understandability framework presented in Sect. 5.

As described in Sect. 5.2.2, we asked participating subjects to voice their thoughts. For the investigation of RQ 2.1, we transcribed the recorded audio files and analyzed how subjects handled the question in which they were asked to describe the processes' behavior, cf. Fig. 7b. The analysis showed that, regardless of whether sub-processes were present or not, subjects described the process in the order activities were supposedly executed, i.e., tried to describe

the process in a *sequential way*. Hence, as first step, subjects skimmed over the process model to find an *entry point* where they could start with describing the process: "... *Ok, this is the, this is the first activity because it has this init constraint*...". Interestingly, subjects seemed to appreciate when a clear starting point for their explanations could be found: "... *it is nice that we have an init activity, so I can start with this*...". A declarative process model, however, does not necessarily have an unique entry point, apparently causing confusion: *"Well... gosh... I've got no clue where to start in this model*..."[4]. After having identified an entry point, subjects tried to figure out in which order activities are to be executed: *"And after given duties to the apprentices there should come these two tasks*...". Finally, subjects indicated where the process supposedly ends: "... *the process ends with the activity give lessons*...."

The sequential way of describing the process models is rather surprising, as it is known that declarative process models rather convey circumstantial information, i.e., overall conditions that produce an outcome, than sequential information, i.e., how the outcome is achieved [14,15]. In other words, in an imperative model, sequences are made explicit, e.g., through sequence flows in BPMN. In a declarative process model, however, such information might not be available at all. For instance, the coexistence constraint [33] defines that two activities *must* occur in the same process instance (or do not occur at all)—the ordering of the activities is not prescribed. As subjects still rather talked about declarative process models in a sequential manner, it appears as if they preferred this kind of information. Interestingly, similar observations could be made in a case study investigating declarative process modeling [62]. Therein, sequential information, such as *"A before B"* or *"then C"*, was preferred for communication.

With respect to this work, the question is in how far sub-processes can support analysts in making sense of the process model. Given that analysts apparently seek for a sequential way of describing the process model, it seems likely that the task of describing a model gets harder for large models, as the analyst cannot just follow sequence flows as in BPMN models, but has to infer which activity could be executed next. Hence, the more activities are present, the more possibilities the analyst has to rule out. Conversely, sub-processes reduce the number of activities per (sub-)model, hence simplifying this task. In order to see whether empirical evidence for this claim could be found, we analyzed the mental effort required for describing process models. During our analysis, we have seen that each subject showed a different *base-level* of mental effort. Hence, a comparison of absolute values of mental effort will be influenced by different base levels. To cancel out this influence and to make mental effort comparable between subjects, we base our analysis on the *relative mental effort*, i.e., the mental effort expended for answering a question divided by the *average mental effort* expended for answering a question about a process model. Thus, for instance, a value of 0.78 indicates that the subject expended 78 % of the average mental effort. Contrariwise, a value of 2.00 indicates that the task was twice as hard as an average task in terms of mental effort.

When comparing the relative mental effort required for describing flat models ($M = 1.68, SD = 0.72$) and hierarchical models ($M = 1.63, SD = 0.72$), however, differences turned out to be marginal (0.05). Nevertheless, this result does not contradict the assumption that sub-processes can improve understanding. Rather, we postulated that mental effort will be lower for *large* process models. Indeed, if the same analysis is performed for the larger models (Process 3 and Process 4), the difference with respect to relative mental effort between flat models ($M = 1.93, SD = 0.93$) and hierarchical models ($M = 1.55, SD = 0.37$) increases to 0.38, i.e., hierarchical models are *easier* to understand. Likewise, for small models, the difference between flat models ($M = 1.43, SD = 0.28$) and hierarchical models ($M = 1.72, SD = 0.50$) increases to -0.29, i.e., hierarchical models are *harder* to understand. These findings are in-line with the framework presented in Sect. 5.1: While large models apparently benefit from information hiding, small models are rather impaired by fragmentation.

So far, we discussed how sub-processes influence analysts in establishing an understanding of a declarative process model. In the following, we investigate in how far the recognition of patterns can support the process analyst. To this end, we will now turn to results obtained from Process 3. Process 3 captures procedures from a company selling electronic devices[5]: After having completed initial tasks, employees either supervise apprentices, handle incoming goods or deal with customer complaints—in the hierarchical model, these three procedures are modeled as sub-processes. Unsurprisingly, all subjects that received the hierarchical model recognized these sub-processes. Interestingly, also *all* subjects that received the flat model described the same sub-processes. However, in contrast to subjects that received hierarchical models, it took them considerably longer to understand that the model could be partitioned this way. In order to visualize this relation, we assessed at which point in time subjects mentioned those sub-processes for the first time. In order to

[4] We allowed subjects to choose their preferred language in order to avoid unnecessary language barriers. The original quote was uttered in Tyrolean dialect: "jå Oiski! Poiski! Då woas ma jå nit wo ånfangn bei dem bledn Modell ...". To improve the comprehensibility of the paper, we translated the quote to English

[5] Due to size, the process models cannot be reproduced here meaningfully, but can be accessed through: http://bpm.q-e.at/experiment/HierarchyDeclarative.

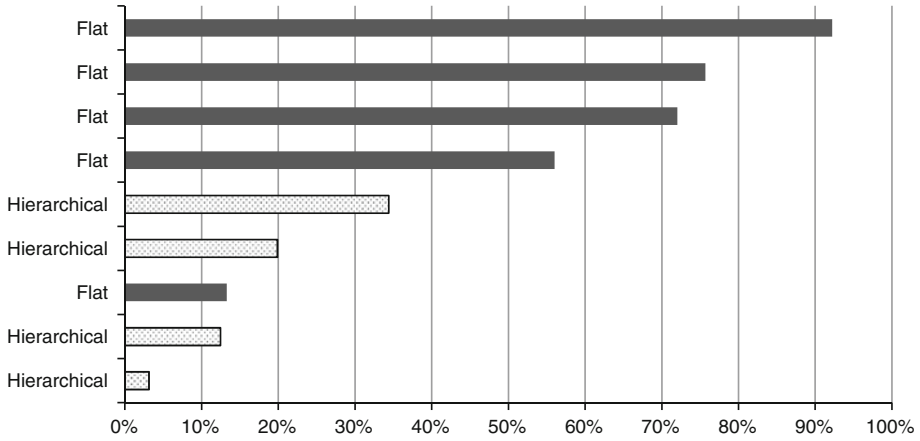


Fig. 9 Duration until first mentioning of sub-processes

eliminate fluctuations such as talking speed, we refrained from looking into absolute duration. Rather, we computed the ratio of the time needed for recognizing the sub-processes divided by the total duration spent for describing the process model. As illustrated in Fig. 9, subjects confronted with the flat model tended to recognize the sub-processes for the first time toward the end of the task only, while subjects confronted with hierarchical models recognized the sub-processes earlier. In particular, for flat models, subjects mentioned sub-process after having expended 62 % of the total time. For hierarchical models, the average ratio dropped to 17 %.

Even though the data indicate that sub-processes could be identified earlier, the question remains why sub-processes were not identified *immediately*. The answer to this question can be found in the way subjects described the process models: *All* subjects described the process in the order activities were supposedly executed. As the sub-processes were to be executed after some initial tasks were performed, subjects first described the initial tasks and *then* the sub-processes. Still, two different patterns could be observed. Subjects who received the hierarchical models mentioned the sub-processes and then described their content. Subject who received flat models rather described the entire model first and toward the end stated that they think that the model could actually be split according to these sub-processes.

Obviously, it is not surprising that subjects mentioned sub-processes earlier in hierarchical models as sub-processes have been explicitly represented. However, when looking into mental effort, similar observations can be made. For flat models, a relative mental effort of 2.00 (200 %) was computed, and for hierarchical models, it dropped to 1.53 (153 %)—providing further evidence that hierarchy was beneficial in this case.

Even though these observations provide empirical evidence for the positive influence of pattern recognition for Process 3, no pattern recognition could be found in Processes 1, 2, and 4. As indicated in the first part of this research question, the size of a model has an impact on whether hierarchy is helping or rather interfering. Likewise, it can be expected that a certain model size is required for pattern recognition, explaining why no effects could be found for Process 1 and Process 2. This, however, does not explain why subjects did not identify sub-processes in Process 4—a potential explanation for this difference can be found in its structure. In particular, the process is to a large extent modeled with precedence constraints, i.e., a constraint that restricts the *ordering* of activities. Hence, subjects could use these constraints to move through the process model in a sequential way. For Process 3, however, such a behavior was not possible, as also constraints that did not convey any sequential information at all (e.g., the not coexistence constraint [33]) were used. Hence, subjects were forced to approach the process model differently. Apparently, the strategy was to divide the process model into parts that could be tackled sequentially—resulting in the described sub-processes.

Furthermore, in Sect. 4.1, we discussed that sub-processes need to relate to a certain *intention*. Indeed, subjects who identified sub-processes in Process 3 described them rather in terms of intentions than on the basis of structure: "*. . . here in this part is about, uhm, managing the apprentices works and also giving the duties . . . this part here is about . . . ah, checking the quality of the good, the incoming good.*". Hence, this may be an additional reason why sub-processes were only identified in Process 3. Apparently, particular activities of Process 3 shared a common intention, e.g., *"checking the quality of the good"*, making them amenable for being extracted as a sub-process. For Process 1, 2, and 4, it can be assumed that such intentions were not given or were not recognized by the subjects.

To summarize, the collected data indicate that sub-processes appear to negatively influence the overall understanding of rather small hierarchical declarative process models, but improve understanding if model size increases. In addition, subjects seemed to approach process models in a sequential manner. When this was not possible, subjects

Table 3 Descriptive statistics of abstraction questions

		Mental effort			Accuracy		
		Min.	Max.	Avg.	Min.	Max.	Avg.
Process 1	Flat	0.84	1.11	0.97	1.00	1.00	1.00
	Hierarchical	0.80	1.13	1.01	0.75	1.00	0.94
Process 2	Flat	0.91	1.07	0.98	1.00	1.00	1.00
	Hierarchical	0.84	1.00	0.90	1.00	1.00	1.00
Process 3	Flat	1.03	1.29	1.14	1.00	1.00	1.00
	Hierarchical	0.92	1.10	1.01	0.75	1.00	0.94
Process 4	Flat	1.08	1.14	1.10	0.75	1.00	0.94
	Hierarchical	1.00	1.03	1.01	0.75	1.00	0.95

apparently tried to divide the process model in manageable, potentially sequential chunks. For hierarchical models, these divisions could directly be perceived in form of sub-processes, hence further supporting the overall understanding of the process model.

5.2.6 RQ 2.2: Does information hiding support analysts in understanding process models?

Besides fostering the recognition of patterns, we argued that information hiding, i.e., using sub-processes to abstract from their content, will support analysts (cf. Fig. 6). In particular, removing information irrelevant for conducting the task at hand will presumably result in a lower mental effort and consequently in higher performance. To investigate this claim, we elaborated questions that could be answered without looking into sub-processes. For instance, consider the declarative process model from Fig. 4 and the following question: *"Must 'Complete writing paper' be executed before 'Get acceptance' can be executed?"*. To answer this question, it is sufficient to consider activities *Complete writing paper*, *Submit paper*, and *Get acceptance* as well as constraints connecting those activities, i.e., 3 activities and 3 constraints. In particular, the analyst can infer that those activities are connected by (chained) succession constraints [33], hence the answer is *yes*. For answering the question in the flat model (cf. Fig. 3), also all constraints and activities describing the relationship between *Complete writing paper* and *Get acceptance* have to considered. Hence, in this case, 4 activities and 6 constraints are of concern. In terms of the framework from Sect. 5.1, such questions will presumably benefit from abstraction, as model elements are hidden in sub-processes, but will not be impaired by fragmentation, as it is not necessary to look into any sub-process for answering the question. Consequently, such questions should be easier to answer in the hierarchical model, resulting in a lower mental effort and higher accuracy, i.e., percentage of correct answers. In order to investigate this research question, we first approach it from a quantitative angle, i.e., we analyze the mental effort and accuracy of questions. Then, we take a qualitative point of view and inspect the think-aloud protocols for evidence of information hiding.

The relative mental efforts for abstraction questions are summarized in Table 3. Except for Process 1, the relative mental effort was always higher in the flat model. To test for statistical significance, we compared the average mental effort for all process models, giving us 36 data points (9 subjects $\times$ 4 models). The applied t-test between questions asked for flat models ($M = 1.05, SD = 0.11$) and questions asked for hierarchical models ($M = 0.98, SD = 0.09$) indicated significant differences: $t(34) = 2.10$, $p = 0.043$, with higher mental effort for questions asked for flat models. With respect to accuracy, i.e., the amount of correctly answered questions, Table 3 provides less conclusive evidence. In fact, accuracy is identical for Process 2, almost identical for Process 4 and higher for Process 1 and Process 3. Unsurprisingly, the applied t-test between questions asked for flat models ($M = 0.99, SD = 0.06$) and questions asked for hierarchical models ($M = 0.96, SD = 0.10$) does *not* indicate significant differences: $t(28.24) = 1.05, p = 0.30$, with lower accuracy for hierarchical models. Summarizing, empirical evidence for the positive influence of information hiding on mental effort could be provided, whereas the influence on accuracy remains less clear.

Summarizing, the data indicate that *information hiding* decreases mental effort—however, no positive influence with respect to accuracy could be observed. Knowing that effects can be considered to be strong when statistically significant for small samples [46] and that mental effort and accuracy have been shown to correlate [64], it seems surprising that no statistical significant differences with respect to accuracy could be found. In the following, we will discuss two potential explanations for this seemingly contradictory situation. First, the high overall accuracy (0.97) and the low standard deviation (0.08) indicate that the lack of significant differences could be attributed to the *ceiling effect* [57]. In other words, the questions were not hard enough or the models were too small to cause a substantial amount of errors,

resulting in low fluctuations of accuracy. In fact, the average mental effort was 3.43, i.e., between *Low mental effort* and *Neither high nor low mental effort*. Second, it has been argued that mental effort is a more sensitive measure than accuracy [64]. Likewise, larger samples are required to show statistical significant differences. Thus, it seems likely that the lack of significant differences with respect to accuracy can be traced back to the rather low sample size (36 data points) and the low complexity of tasks.

Up to now we focused on quantitative data to investigate the influence of information hiding. In the following, we turn to the think-aloud protocols and video recordings, discussing qualitative evidence for the utilization of information hiding. In particular, regardless of whether sub-processes were present or not, a two-step procedure could be observed. In the first step, subjects identified all activities relevant for answering a question. Apparently depending on personal preference, subjects used a pencil to highlight these activities or simply placed a finger on the paper. In cognitive psychology, this is referred to as *external memory* [65]. The information, which activities have to considered for answering the question, is stored externally instead of taking up the human mind's working memory. In the second step, subjects performed the reasoning, i.e., interpreted the constraints relevant for these activities. Interestingly, after step 1 was performed, we could observe subjects actively pursuing information hiding. In particular, in hierarchical models, sheets of papers that contained irrelevant sub-processes for the question at hand were removed, e.g., *"I don't need this here I think. . .."* A similar pattern could be observed in the flat models: After having identified which parts of the model are relevant for answering the question at hand, subjects followed various strategies for hiding irrelevant information. For instance, a hand was used to cover up irrelevant parts of the model ("*. . . this part of the model cannot be performed. . .*") or the relevant part of the models was highlighted: "*. . . cannot occur, since I've got here some kind of partial process. . .*"[6]. Hence, we conclude that information hiding appears to be a strategy that is intuitively followed by subjects. Interestingly, also for flat models, where all information is present at once, subjects emulated information hiding by covering up irrelevant parts of the model. Still, as indicated in Table 3, information hiding seems to be rather present in hierarchical models than in flat models.

5.2.7 RQ 3: Does fragmentation hinder analysts in understanding Process models?

After having provided empirical evidence that analysts are basically able to understand hierarchy (RQ 1) and positive influence of sub-processes (RQ 2.1 and RQ 2.2), now we turn to the postulated negative influence. As argued in Sect. 5.1, tasks that involve the content of several sub-processes require the analyst to mentally integrate these sub-processes, imposing a higher mental effort and leading to lower performance. In order to empirically investigate this claim and similar to RQ 2.2, we elaborated questions that presumably do *not* benefit from abstraction, but suffer from fragmentation. Hence, such questions should be easier to be answered in a flat model, as they are not negatively influenced by hierarchy. More precisely, questions answered in the hierarchical model should require a higher mental effort and have lower accuracy. Considering Fig. 4, such a questions could be: *"Is 'Work on revision' executable after 'Get acceptance' was executed?"*. To answer this question, the analyst has to locate *Work on revision* within sub-process *Revise paper*. Then, the analyst has to infer that the negation response constraint [33] between *Get acceptance* and *Revise paper* also affects the execution of *Work on revision*. Hence, *Work on revision* cannot be executed after *Get acceptance*. In the flat model (cf. Fig. 3), *Work on revision* and *Get acceptance* are directly connected by a negation response constraint. Thus, it is sufficient to interpret the meaning of a single constraint. Similar to RQ 2.2, we start by approaching RQ 3 from a quantitative angle and take a qualitative point of view afterward.

The analysis of results follows the same strategy as applied in RQ 2.2, i.e., we computed the relative mental effort and accuracy for all models. As can be see in Table 4, the data indicate a higher average mental effort for questions that were asked in the hierarchical model ($M = 1.02, SD = 0.10$) than for flat models ($M = 0.95, SD = 0.11$). In particular, the average mental effort is higher in all hierarchical models, except for Process 1. As in RQ 2.2, we employed a t-test to test for statistical significance: $t(34) = -2.10$, $p = 0.043$, with higher mental effort for hierarchical models. For accuracy, the picture is less clear. For Process 1 and Process 3, the accuracy was higher in the flat model, in Process 2, differences are marginal, whereas in Process 4, accuracy was lower in the flat version. Likewise, also the applied t test between questions asked in hierarchical models ($M = 0.89, SD = 0.15$) and questions asked in flat models ($M = 0.96, SD = 0.10$) was not significant: $t(28.46) = 1.63$, $p = 0.12$, with lower accuracy for hierarchical models. Hence, similar to RQ 2.2, we could provide empirical evidence for the negative influence of hierarchy on mental effort, but could not show differences with respect to accuracy.

Interestingly, a similar pattern of results as described in RQ 2.2 could be observed. Again, mental effort was significantly different, while no significant differences with respect to accuracy could be shown. In RQ 2.2, we argued that these results were to a certain extent caused by the ceiling effect, i.e., high accuracy and low standard deviation. In RQ 3, further evidence for this assumption is provided. More specifi-

[6] Original quote: "…cannot occur, da ich hier so'n Teilprozess hab. . .".

Table 4 Descriptive statistics of fragmentation questions

		Mental effort			Accuracy		
		Min.	Max.	Avg.	Min.	Max.	Avg.
Process 1	Flat	0.89	1.16	1.03	0.75	1.00	0.95
	Hierarchical	0.87	1.20	0.99	0.75	0.75	0.75
Process 2	Flat	0.93	1.09	1.02	0.75	1.00	0.94
	Hierarchical	1.00	1.18	1.10	0.75	1.00	0.95
Process 3	Flat	0.71	0.97	0.86	1.00	1.00	1.00
	Hierarchical	0.90	1.08	0.99	0.50	1.00	0.81
Process 4	Flat	0.86	0.92	0.90	0.75	1.00	0.94
	Hierarchical	0.97	1.00	0.99	1.00	1.00	1.00

cally, the mean accuracy was lower (0.92 vs. 0.97), while the standard deviation increased (0.13 vs. 0.08). In line with these changes, also the p value reported by the t test dropped near significance (0.12 vs. 0.30). Hence, it seems likely that also for RQ 3 lack of significant differences with respect to accuracy can be traced back to sample size and low complexity of tasks. Nevertheless, empirical evidence for the negative influence of hierarchy in terms of mental effort could be provided.

In order to enhance RQ 3 with qualitative insights, we examined the think-aloud protocols for evidence of fragmentation. A particularly explicit case of fragmentation can be found in question 4 from Process 2. Here, subjects were asked to answer how often *Decide on teaching method*, contained in sub-process *Prepare lessons*, could be executed. *Decide on teaching method* was constrained to be executed exactly once in the sub-process *Prepare lessons*. *Prepare lessons*, in turn, was also restricted to be executed exactly once. Hence, subjects had to combine these two constraints to find out that *Decide on teaching method* could be executed exactly once. The reasoning process required to establish this answer can be found in a subject's think-aloud protocol: "*. . .yes, has to be executed exactly once . . . it is in this sub-process of prepare lessons. Prepare lessons has to be executed exactly once and also in the sub-process exactly once. One times one is one. . .*"[7]. As described in RQ 2.2, subjects first located relevant activities and then interpreted associated constraints. In this particular case, the subject understood that it had to combine the selection constraint on *Decide on teaching method* with the selection constraint on *Prepare lessons*, i.e., had to integrate these two selection constraints. Even though this integration task appears especially easy (*"one times one is one"*), it emphasizes the problem of fragmentation: It requires the analyst to combine the semantics of (potentially) several constraints. This, in turn, has been shown to be the major reason for misinterpreting declarative process models (cf. RQ 1), providing further empirical evidence for the negative influence of fragmentation.

An apparently especially difficult integration task can be found in a *fragment* of Process 1, cf. Fig. 10. In particular, the subjects had to assess the statement *"'Write code' has to be executed before 'Merge fix' can be executed."*. To this end, three facts have to be combined. First, *Write code* is contained in sub-process *Apply TDD*, while *Merge fix* can be found in sub-process *Work with production software*. Second, *Apply TDD* and *Work with production software* are connected by a precedence constraint, hence *Apply TDD* must be executed before *Work with production software* can be executed. Hence, it could mistakenly be inferred that *Write code* must be executed before *Merge fix* can be executed. However, third, *Write code* is not necessarily executed when *Apply TDD* is executed. Rather, *Write test* must be executed at least once and consequently also *Run tests* must be executed at least once due to the chained response constraint [33] between these two activities. *Write code*, though, is not required—hence, *Merge fix* can also be executed without *Write code* being performed before.

For illustration purpose, consider the following excerpt from a think-aloud transcript: *"Write code has to be, write code, where are you, here, has to be executed before merge fix can be executed."*. Here the subject searches for activities *Write code* and *Merge fix*. Then, the subject examines the relationship between the sub-processes which contain these activities: *"Yes, because before, ahm, before work with production software which is the sub-process where merge fix is . . . apply TDD has to be performed before."*. Here, the subject apparently falsely integrates the precedence constraint between *Apply TDD* and *Work with production software* with the activities contained therein. Knowing that the subject answered 29 out of 32 (91 %) ques-

[7] Original quote: "…ja, muss immer genau einmal ausgeführt werden, das is in dem, es is in dem Subprozess von prepare lessons. Prepare lessons muss genau einmal ausgeführt werden und das muss in dem Subprozess genau einmal, und ein mal eins ergibt bei mir auch wieder eins. . .".

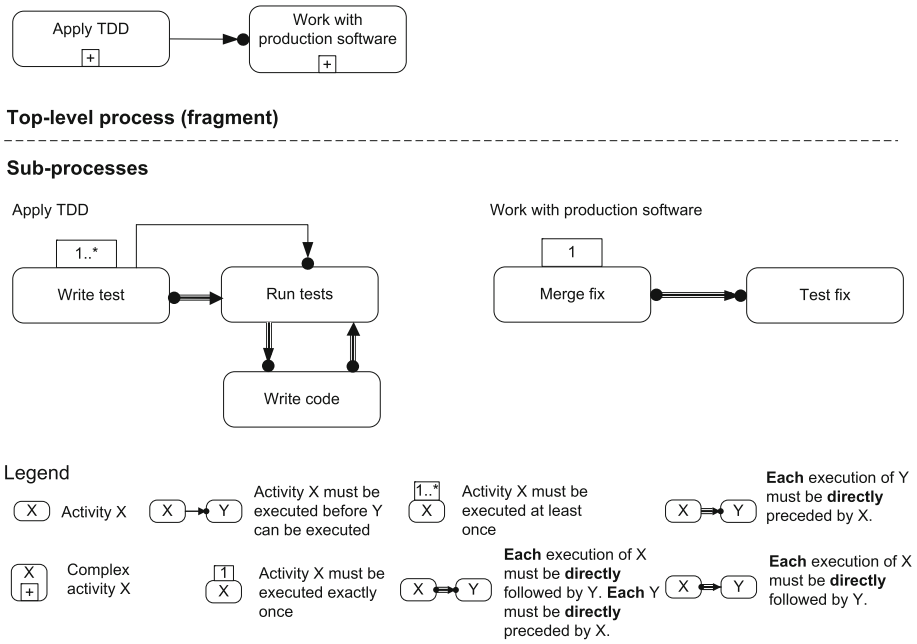


Fig. 10 Declarative process model with difficult integration task

tions correctly, it can be assumed that the subject tried its best to answer the questions correctly. Hence, we conclude that this task indeed posed a significant challenge for the subject.

5.3 Discussion

The presented results show a diversified picture of hierarchy in declarative models. Basically, the findings of RQ 1 indicate that analysts are able to properly interpret sub-processes. However, the adoption of sub-processes does not necessarily improve the understandability of a model. While pattern recognition (RQ 2.1) and information hiding (RQ 2.2) may lower the mental effort for understanding a process model, fragmentation (RQ 3) appears to impose an additional burden on the analyst.

Besides providing empirical support for the understandability framework proposed in Sect. 5.1, the results indicate that the benefits of a hierarchization depend on which kind of information should be extracted. In other words, if the question an analyst is interested in, rather benefits from abstraction than being impaired by fragmentation, understandability will presumably improve. Contrariwise, if fragmentation prevails, the model will presumably become more difficult to understand. Thus, it seems worthwhile to maximize the ratio of abstraction to fragmentation. In this sense, *dynamic process visualizations* [1,22,43] seem to be promising, as they allow to visualize the process model according to the analyst's demands. In the context of this work, such a dynamic visualization would ensure that all relevant modeling elements are visible, while irrelevant modeling elements are hidden in sub-processes. This, however, would require an automated restructuring of hierarchical declarative process models. Such techniques, however, are not in place yet and only possible for process models that do not make use of enhanced expressiveness (cf. Sect. 4.2).

Hence, for the time being, analysts will have to rely on *statically* visualized process models, as used in this work. For the interpretation of such models, we could identify different strategies in the think-aloud protocols and video material. Basically, analysts appear to approach declarative process models in a sequential manner, i.e., they tend to describe the process in the ordering activities can be executed. Knowing that imperative process modeling languages, e.g., BPMN, are much wider spread than declarative process modeling languages, one might argue that this indicates that subjects were biased by the former category of modeling languages. On the other hand, it was found that domain experts, i.e., persons unfamiliar with business process modeling, were also inclined toward sequential behavior [62]. Hence, it seems likely that the abstract nature of declarative process models does not naturally fit the human way of reasoning. Evidence that constraints indeed may pose a significant challenge for the analyst could be found in the tasks where subjects were asked to describe a process model. Therein, we could find indications that for the larger process models, sub-processes helped to divide the model into manageable parts, i.e., the number of interacting constraints seems to play an essential role. Further evidence for this thesis is provided by the finding that subjects intuitively sought to reduce the number of constraints by, e.g., putting away sheets describing irrelevant sub-processes or, in a flat model, using the hand to hide irrelevant parts of the model. Unsurprisingly, it has been shown that relieving analysts from interpreting constraints supports

the maintenance [64,67] and execution of declarative process models [61].

In this work, we have not considered the granularity of modularizations. Likewise, we have not investigated whether correct levels of abstraction have been applied for sub-processes, as discussed in detail in [11,41]. Rather, our work has to be seen as an orthogonal perspective to these aspects. Even when optimizing granularity and abstraction levels, a process model may be modularized in various ways. The framework proposed in this work may then be used as an additional perspective, helping the analyst to decide for a specific modularization.

Similarly, the results have to been in the light of guidelines for modularization. For instance, according to the *good decomposition model* [58], proper modularization should satisfy *minimality*, *determinism*, *losslessness*, *weak coupling*, and *strong cohesion*. Again, abstraction and fragmentation have to be seen as an additional perspective. Basically, satisfying the conditions of the good decomposition model can be related to optimizing the ratio between abstraction and fragmentation. For instance, achieving *strong cohesion* clearly aims at increasing abstraction by keeping closely related objects together (non-related object will have to be placed in different sub-models to achieve strong cohesion, hence fostering abstraction). *Weak coupling*, in turn, aims at minimizing fragmentation by minimizing connections between sub-models and hence decreasing potential switches between sub-models. *Losslessness*, i.e., that no information is lost when introducing sub-processes, is not captured in our framework, as the focus of our work is put on models rather than on their creation. Finally, achieving *minimality*, i.e., non-redundancy, and *determinism* seem desirable for modularization. However, in our opinion, these factors are not necessarily related to decomposition only, but should be rather seen as general modeling guidelines that also hold for non-modularized models. As our framework specifically focuses on modularization, we do not see a direct connection between our framework.

With respect to empirical investigations of hierarchical models in general, the interplay of positive and negative influences is also of interest. In particular, it doubts in how far results obtained in empirical comparisons between flat and hierarchical models are meaningful if questions have not been designed carefully. More specifically, in this work, significant results with respect to mental effort could be reported for abstraction in RQ 2.2 and for fragmentation in RQ 3. If, however, the distinction between abstraction and fragmentation is not made and comparisons are conducted between flat models ($M = 1.00, SD = 0.12$) hierarchical models ($M = 1.00, SD = 0.09$) only, effects disappear: $t(70) = 0.00, p = 1.00$. Likewise, similar observations could be observed from a qualitative angle. In particular, subjects were found to actively make use of information hiding and remarked that they were lost when model size increased due to lack of sub-processes—indicating the positive influence of hierarchy. Contrariwise, subjects could be observed struggling with combining the semantics of several sub-processes, i.e., struggled with fragmentation. Hence, merely comparing hierarchical models and flat models seems to be too shortsighted. Rather, in the experimental design, positive and negative effects should be distinguished. Against this background, seemingly contradicting results from empirical investigations into hierarchy can be explained in a plausible way. In works reporting from positive influence, e.g., [28,45], questions benefiting from abstraction probably prevailed. In inconclusive works, e.g., [6,49], questions benefitting from abstraction and questions impaired by fragmentation were probably in balance. In works reporting from negative influence, in turn, e.g., [5,9], probably questions impaired by fragmentation prevailed.

6 Limitations

Apparently, several limitations, particularly concerning the empirical investigation, apply to this work. First, the empirical evaluation provides promising results; however, the rather low sample size (9 subjects) is a clear threat to the generalization of results. Second, even though the process models used in this study vary in the number of activities, constraints, and sub-processes, it is not entirely clear whether the obtained results are applicable to every hierarchical declarative process models. In this vein, we have also considered process models with a nesting level of one only, i.e., none of the sub-processes was refined using further sub-processes. As it has been shown that an overuse of sub-processes may negatively impact the understanding of a model [9], the limited nesting level has to be seen as a further limitation of this study. Third, and similarly, the questions used to assess the understandability can only address a limited number of aspects. Even though questions were designed to representatively cover several aspects of models (cf. [24]), a bias favoring certain questions cannot be ruled out entirely. Fourth, all participating subjects indicated academic background, limiting the generalization of results. However, subjects also indicated profound background in BPM; hence, we argue that they can be seen as proxies for professionals. Finally, this work focuses on control-flow aspects of declarative models. Other perspectives of process models, such as resources or data, have not been taken into account yet.

7 Related work

In this work, we discussed characteristics of hierarchy in declarative process models and the impact on understandabil-

ity. The impact of hierarchy on understandability has been studied in various conceptual modeling languages, such as imperative business process models [45], ER diagrams [28,49], and UML statechart diagrams [5,7,8] (an overview is presented in [63]). General considerations about modularization, resulting in the *good decomposition model* [58], have been adopted for the modularization of object-oriented design [2] and the modularization of Event-driven Process Chains [19]. Guidelines for the use of sub-processes in imperative process models are provided in [21,26,48]. Even though these works provide valuable insights into hierarchical models, none of these works deals with declarative process models. The understandability of declarative process models in general has been investigated in [66–68]; however, in contrast to this work, hierarchy is not discussed. With respect to understandability of process models in general, work dealing with the understandability of imperative business process models is related. In [26], modeling guidelines are presented that target to improve the understandability of imperative process models. In particular, it is stressed that the size of a model has *"undesirable effects on understandability and likelihood of errors"* [26] and that *imperative* process models should be decomposed if growing larger than 50 modeling elements, hence emphasizing the need for information hiding through modularization. In how far further imperative-model-specific insights, such as the connector degree, i.e., ingoing and outgoing arcs from connectors, can be transferred to declarative process models, still needs to be investigated. Similarly, in [12,56] the understanding of process models is assessed through the adoption of structural metrics. In [25], the relationship between the size of imperative process models and error rates is established.

In this work, we focused on the outcome of a process modeling endavour, i.e., the process model. Recently, researchers have also began to investigate the process of creating a process model, referred to as *the process of process modeling* [39]. Similar to this work, the way how analysts make sense of a process model while creating it is investigated—for instance, by visualizing the process of process modeling [3]. Similarly, different personalized modeling styles [38] and modeling strategies have been identified [4]. Even though all these works focus on an imperative modeling language, i.e., a subset of BPMN, similar investigations into declarative process modeling languages seem promising. Likewise, also methodological considerations of how to come up with a proper modularization are related. For instance, in [11] a development method, which foresees the creation of modularized business processes, is described.

Besides assessing the understandability of modularization, several authors investigated potential ways of automatically creating modularized models. In particular, in [40], an approach for automatically aggregating activities based the most relevant activities of a process model is proposed. Similarly, in [51], an approach for the automated abstraction of control flow, based on behavioral profiles, is described. Another automated approach for modularization is described in [50]—here, meronymy relations between activity labels are employed to automate modularization. Even though these approaches promise to provide abstraction in an automated way, it is unclear in how far the created models will be understandable to the end-user, which is of concern in this work. Further, the application to declarative process models are not discussed.

Finally, in [27,33], technical aspects of declarative business process models, such as the definition of modeling languages or verification of models, are investigated. In contrast to this work, understandability aspects are neglected and the unique semantics and expressiveness enabled by sub-processes are not elaborated.

8 Summary and outlook

In this work, we examined hierarchy in declarative business process models. After elaborating on the semantics, we discussed the usage and peculiarities of hierarchy. In particular, we showed that hierarchy enhances expressiveness, but cannot be used arbitrarily to any model fragment. Subsequently, we discussed implications on the understandability of declarative process models. Thereby, we built upon previous work and proposed a cognitive-theory-based framework to systematically assess the impact of hierarchy on understandability in declarative process models. In general, it can be said that hierarchy should be handled with care. On the one hand, information hiding and increased pattern recognition promise gains in terms of understandability. On the other hand, the integration of constraints and switching between sub-processes may compromise the understandability of respective models. The empirical investigation testing these claims followed an approach combining quantitative and qualitative research methods. Speaking in terms of quantitative data, support for the postulated influence on mental effort could be found. For accuracy, however, results are rather inconclusive. In addition, qualitative data, i.e., think-aloud protocols and video recordings, provided valuable insights into the reasoning processes of analysts. All in all it can be said that the collected data provide empirical evidence that experiments investigating hierarchical models need to be designed with care, as the type of question asked can have a significant influence on the outcome.

More generally, this work contributes to a more systematic assessment of hierarchy in declarative business process models. In particular, it provides deeper insights into how far sub-processes influence the understanding of a declarative process model. These findings, in turn, foster the development of the guidelines for the adoption of hierarchy that

are based on empirical evidence. Similarly, we hope that this work contributes to more objective discussions about the proper use of sub-processes, as it provides criteria for arguing about the influence a certain hierarchical structure.

Even though the data collected in this work corroborated the postulated influence of hierarchy on understandability, further investigations are desirable. In particular, further replications as well are more complex process models seem to be appropriate means for additional empirical tests. Although the think-aloud protocols already provided a detailed view on analyst's reasoning processes, we plan to additionally employ eye movement analysis [37] for even more detailed analysis. Based on these insights, we intend to further develop this work toward empirically founded guidelines for the design of hierarchical declarative process models.

References

1. Bobrik, R., Reichert, M., Bauer, T.: Requirements for the visualization of system-spanning business processes. In: Proceedings of the DEXA'05, pp. 948–954 (2005)
2. Burton-Jones, A., Meso, P.N.: Conceptualizing systems for understanding: an empirical test of decomposition principles in object-oriented analysis. ISR **17**(1), 38–60 (2006)
3. Claes, J., Vanderfeesten, I., Pinggera, J., Reijers, H., Weber, B., Poels, G.: Visualizing the Process of Process Modeling with PPM-Charts. In Proceedings of the TAProViz '12, pp. 744–755 (2013)
4. Claes, J., Vanderfeesten, I., Reijers, H., Pinggera, J., Weidlich, M., Zugal, S., Fahland, D., Weber, B., Mendling, J., Poels, G.: Tying process model quality to the modeling process: the impact of structuring, movement, and speed. In: Proceedings of the BPM '12, pp. 33–48 (2012)
5. Cruz-Lemus, J., Genero, M., Piattini, M.: Using controlled experiments for validating uml statechart diagrams measures. In: Software Process and Product Measurement of LNCS, vol. 4895, pp. 129–138. Springer/Heidelberg (2008)
6. Cruz-Lemus, J., Genero, M., Piattini, M., Toval, A.: Investigating the nesting level of composite states in uml statechart diagrams. In: Proceedings of the QAOOSE '05, pp. 97–108 (2005)
7. Cruz-Lemus, J.A., Genero, M., Manso, M.E., Morasca, S., Piattini, M.: Assessing the understandability of UML statechart diagrams with composite states–A family of empirical studies. Empir. Softw. Eng. **25**(6), 685–719 (2009)
8. Cruz-Lemus, J.A., Genero, M., Morasca, S., Piattini, M.: Using practitioners for assessing the understandability of UML statechart diagrams with composite states. In: Proceedings of the ER Workshops '07, pp. 213–222 (2007)
9. Cruz-Lemus, J.A., Genero, M., Piattini, M., Toval, A.: An empirical study of the nesting level of composite states within uml statechart diagrams. In: Proceedings of the ER Workshops, pp. 12–22 (2005)
10. Damij, N.: Business process modelling using diagrammatic and tabular techniques. Bus. Process Manag. J. **13**(1), 70–90 (2007)
11. de Weger, M.K.: Structuring of business processes: an architectural approach to distributed systems development and its application to business processes. PhD thesis, University of Twente (1998)
12. Dumas, M., Rosa, M.L., Mendling, J., Mäesalu, R., Reijers, H.A., Semenenko, N.: Understanding business process models: the costs and benefits of structuredness. In: Proceedings of the CAiSE '12, pp. 31–46 (2012)
13. Ericsson, K.A., Simon, H.A.: Protocol Analysis: Verbal Reports as Data. MIT Press, Cambridge (1993)
14. Fahland, D., Mendling, J., Reijers, H., Weber, B., Weidlich, M., Zugal, S.: Declarative versus imperative process modeling languages: the issue of maintainability. In: Proceedings of the ER-BPM '09, pp. 65–76 (2009)
15. Fahland, D., Mendling, J., Reijers, H.A., Weber, B., Weidlich, M., Zugal, S.: Declarative versus imperative process modeling languages: the issue of understandability. In: Proceedings of the EMM-SAD '09, pp. 353–366 (2009)
16. Goguen, J.A., Varela, F.J.: Systems and distinctions; duality and complementarity. Int. J. Gen. Syst. **5**(1), 31–43 (1979)
17. Gopher, D., Brown, R.: On the psychophysics of workload: why bother with subjective measure? Human Fact. J. Human Fact. Ergon. Soc. **26**(5), 519–532 (1984)
18. Haisjackl, C.: Test driven modeling meets declarative process modeling a case study. University of Innsbruck, August, Master's thesis (2012)
19. Johannsen, F., Leist, S.: Wand and Weber's Decomposition Model in the Context of Business Process Modeling. BISE **4**(5), 271–286 (2012)
20. Khatri, V., Vessey, I., Ramesh, P.C.V., Park, S.-J.: Understanding conceptual schemas: exploring the role of application and IS domain knowledge. Inf. Syst. Res. **17**(1), 81–99 (March 2006)
21. Kock, N.F.: Product flow, breadth and complexity of business processes: an empirical study of 15 business processes in three organizations. Bus. Process Re-eng. Manag. J. **2**(2), 8–22 (1996)
22. Kolb, J., Kammerer, K., Reichert, M.: Updatable process views for user-centered adaption of large process models. In: Proceedings of the ICSOC'12, pp. 484–498 (2012)
23. Larkin, J.H., Simon, H.A.: Why a diagram is (sometimes) worth ten thousand words. Cogn. Sci. **11**(1), 65–100 (1987)
24. Melcher, J., Seese, D.: Towards validating prediction systems for process understandability: measuring process understandability. In: Proceedings of the SYNASC '08, pp. 564–571 (2008)
25. Mendling, J.: Detection and Prediction of Errors in EPC Business Process Models. PhD thesis, Vienna University of Economics and Business Administration (2007)
26. Mendling, J., Reijers, H.A., van der Aalst, W.M.P.: Seven process modeling guidelines (7pmg). Inf. Softw. Technol. **52**(2), 127–136 (2010)
27. Montali, M.: Specification and Verification of Declarative Open Interaction Models. Springer, Berlin (2010)
28. Moody, D.L.: Cognitive load effects on end user understanding of conceptual models: an experimental analysis. In: Proceedings of the ADBIS '04, pp. 129–143 (2004)
29. OMG: BPMN version 2.0. http://www.omg.org/spec/BPMN/2.0/PDF/ (2011)
30. Paas, F., Renkl, A., Sweller, J.: Cognitive load theory and instructional design: recent developments. Educ. Psychol. **38**(1), 1–4 (2003)
31. Parnas, D.L.: On the criteria to be used in decomposing systems into modules. Commun. ACM **15**(12), 1053–1058 (1972)
32. Parsons, J., Cole, L.: What do the pictures mean? Guidelines for experimental evaluation of representation fidelity in diagrammatical conceptual modeling techniques. DKE **55**(3), 327–342 (2005)
33. Pesic, M.: Constraint-Based Workflow Management Systems: Shifting Control to Users. PhD thesis, TU Eindhoven (2008)

34. Pesic, M., Schonenberg, H., Sidorova, N., van der Aalst, W.: Constraint-based workflow models: change made easy. In: Proceedings of the CoopIS '07, pp. 77–94 (2007)
35. Pesic, M., Schonenberg, H., van der Aalst, W.: DECLARE: full support for loosely-structured processes. In: Proceedings of the EDOC '07, pp. 287–298 (2007)
36. Pichler, P., Weber, B., Zugal, S., Pinggera, J., Mendling, J., Reijers, H.: Imperative versus declarative process modeling languages: an empirical investigation. In: Proceedings of the ER-BPM '11, pp. 383–394 (2012)
37. Pinggera, J., Furtner, M., Martini, M., Sachse, P., Reiter, K., Zugal, S., Weber, B.: Investigating the process of process modeling with eye movement analysis. In: Proceedings of the ER-BPM '12, pp. 438–450 (2013)
38. Pinggera, J., Soffer, P., Zugal, S., Weber, B., Weidlich, M., Fahland, D., Reijers, H., Mendling, J.: Modeling styles in business process modeling. In: Proceedings of the BPMDS '12, pp. 151–166 (2012)
39. Pinggera, J., Zugal, S., Weidlich, M., Fahland, D., Weber, B., Mendling, J., Reijers, H.: Tracing the process of process modeling with modeling phase diagrams. In: Proceedings of the ER-BPM '11, pp. 370–382 (2012)
40. Polyvyanyy, A., Smirnov, S., Weske, M.: Process model abstraction: a slider approach. In: Proceedings of the EDOC '08, pp. 325–331 (2008)
41. Quartel, D.: Action relations. Basic design concepts for behaviour modelling and refinement. PhD thesis, University of Twente (1998)
42. Quartel, D.A., Pires, L.F., van Sinderen, M.J., Franken, H.M., Vissers, C.A.: On the role of basic design concepts in behaviour structuring. Comput. Netw. ISDN Syst. **29**(4), 413–436 (1997)
43. Reichert, M., Kolb, J., Bobrik, R., Bauer, T.: Enabling personalized visualization of large business processes through parameterizable views. In: Proceedings of the SAC'12, pp. 1653–1660 (2012)
44. Reichert, M., Weber, B.: Enabling Flexibility in Process-Aware Information Systems: Challenges, Methods, Technologies. Springer, Berlin (2012)
45. Reijers, H., Mendling, J., Dijkman, R.: Human and automatic modularizations of process models to enhance their comprehension. Inf. Syst. **36**(5), 881–897 (2011)
46. Royall, R.M.: The effect of sample size on the meaning of significance tests. Am. Stat. **40**(4), 313–315 (1986)
47. Scaife, M., Rogers, Y.: External cognition: how do graphical representations work? Int. J. Human Comput. Stud. **45**(2), 185–213 (1996)
48. Sharp, A., McDermott, P.: Workflow modeling: tools for process improvement and application development. Artech House (2011)
49. Shoval, P., Danoch, R., Balabam, M.: Hierarchical entity-relationship diagrams: the model, method of creation and experimental evaluation. Req. Eng. **9**(4), 217–228 (2004)
50. Smirnov, S., Dijkman, R., Mendling, J., Weske, M.: Meronymy-based aggregation of activities in business process models. In: Proceedings of the ER '10, pp. 1–14 (2010)
51. Smirnov, S., Weidlich, M., Mendling, J.: Business process model abstraction based on behavioral profiles. In: Proceedings of the ICSOC '10, pp. 1–16 (2010)
52. Soffer, P., Rolland, C.: Combining intention-oriented and state-based process modeling. In: Proceedings of the ER '05, pp. 47–62 (2005)
53. Sweller, J.: Cognitive load during problem solving: effects on learning. Cogn. Sci. **12**(2), 257–285 (1988)
54. Sweller, J., Chandler, P.: Why some material is difficult to learn. Cogn. Instr. **12**(3), 185–233 (1994)
55. van der Aalst, W., ter Hofstede, A.H.M.: YAWL: yet another workflow language. Inf. Syst. **30**(4), 245–275 (June 2005)
56. Vanderfeesten, I., Reijers, H.A., Mendling, J., van der Aalst, W., Cardoso, J.: On a quest for good process models: the cross-connectivity metric. In: Proceedings of the CAiSE '08, pp. 480–494 (2008)
57. Vogt, W.P.: Dictionary of Statistics & Methodology: A Nontechnical Guide for the Social Sciences, 4th edn. SAGE Publications, Beverly Hills (2011)
58. Wand, Y., Weber, R.: An ontological model of an information system. IEEE TSE **16**(11), 1282–1292 (1990)
59. Weber, B., Reichert, M., Mendling, J., Reijers, H.A.: Refactoring large process model repositories. Comput. Indus. **62**(5), 467–486 (2011)
60. Weber, B., Reichert, M., Rinderle, S.: Change patterns and change support features—enhancing flexibility in process-aware information systems. DKE **66**(3), 438–466 (2008)
61. Weber, B., Reijers, H.A., Zugal, S., Wild, W.: The declarative approach to business process execution: an empirical test. In: Proceedings of the CAiSE '09, pp. 270–285 (2009)
62. Zugal, S., Haisjackl, C., Pinggera, J., Weber, B.: Empirical Evaluation of Test Driven Modeling. IJISMD, to appear. Available online
63. Zugal, S., Pinggera, J., Mendling, J., Reijers, H., Weber, B.: Assessing the impact of hierarchy on model understandability–a cognitive perspective. In: Proceedings of the EESSMod '11, pp. 123–133 (2011)
64. Zugal, S., Pinggera, J., Reijers, H., Reichert, M., Weber, B.: Making the case for measuring mental effort. In: Proceedings of the EESSMod '12, pp. 37–42 (2012)
65. Zugal, S., Pinggera, J., Weber, B.: Assessing process models with cognitive psychology. In: Proceedings of the EMISA '11, pp. 177–182 (2011)
66. Zugal, S., Pinggera, J., Weber, B.: Creating declarative process models using test driven modeling suite. In: Proceedings of the CAiSE, Forum '11, pp. 16–32 (2011)
67. Zugal, S., Pinggera, J., Weber, B.: The impact of testcases on the maintainability of declarative process models. In: Proceedings of the BPMDS '11, pp. 163–177 (2011)
68. Zugal, S., Pinggera, J., Weber, B.: Toward enhanced life-cycle support for declarative processes. JSEP **24**(3), 285–302 (2012)
69. Zugal, S., Soffer, P., Pinggera, J., Weber, B.: Expressiveness and understandability considerations of hierarchy in declarative business process models. In: Proceedings of the BPMDS '12, pp. 167–181 (2012)

Author Biographies

Stefan Zugal is a Ph.D. candidate at the University of Innsbruck (Austria). Stefan is a member of Quality Engineering (QE) Research Group and member of the Research Cluster on Business Processes and Workflows at QE. Stefan received his M.Sc. degree from the Department of Computer Science, University of Innsbruck in 2008. His main research interests are declarative business process models and the understandability of business process models. Stefan has published more than 30 refereed papers in international journals, conferences and workshops.

Pnina Soffer is a senior lecturer in the Information Systems Department at the University of Haifa. She received her B.Sc. (1991) and MSc (1993) in Industrial Engineering from the Technion, Ph.D. in Information Systems Engineering from the Technion (2002). Her research deals with business process modelling and management, requirements engineering, and conceptual modelling, addressing issues such as goal orientation, flexibility, interoperability, and context-aware adaptation. Her research has appeared in journals such as Journal of the AIS, European J of IS, Requirements Engineering, Information Systems, and others. She has served as a guest editor of a number of special issues related to various business process topics such as business process flexibility, coordinated development of business processes and information systems, and business process design. Pnina has served in program committees of numerous conferences, including CAiSE, BPM, ER, and others. She is a member of the CAiSE steering committee and an organizer of the BPMDS working conference. She is a member of editorial boards of several journals including the Journal of the AIS.

Cornelia Haisjackl is a Ph.D. candidate at the University of Innsbruck (Austria). She is a member of Quality Engineering (QE) Research Group and member of the Research Cluster on Business Processes and Workflows at QE. Cornelia received her M.Sc. degree from the Department of Computer Science, University of Innsbruck in 2012. Her main research interests are declarative business process models and the understandability of business process models.

Jakob Pinggera is a Ph.D. candidate at the University of Innsbruck (Austria). Jakob is a member of Quality Engineering (QE) Research Group and member of the Research Cluster on Business Processes and Workflows at QE. Jakob received his M.Sc. degree from the Department of Computer Science, University of Innsbruck in 2009. His main research interest is the process of process modeling, i.e., how modelers create business process models. His research interests further include business process modeling notations and business process quality. Jakob has published more than 30 refereed papers in international journals, conferences and workshops.

Manfred Reichert is professor at the University of Ulm, Germany and co-director of the Institute of Databases and Information Systems. His major research interests are next generation process management technology (e.g., adaptive processes, process variability, data-driven and object-centric processes, mobile processes), service-oriented computing (e.g., service interoperability, service evolution), and advanced applications for flexible information systems (e.g., e-health and automotive engineering). Together with Peter Dadam he pioneered the work on the ADEPT process management technology. Manfred is co-founder of the AristaFlow GmbH and has been participating in numerous research projects in the BPM area contributing more than 200 scientific papers on BPM-related topics. His book entitled "Enabling Flexibility in Process-Aware Information Systems", which he co-authored with Barbara Weber, was published by Springer in September 2012. Manfred has been PC Co-Chair of the BPM'08, CoopIS'11, and EDOC'13 conferences and General Chair of the BPM'09 conference.

Barbara Weber is associate professor at the University of Innsbruck (Austria). Barbara is a member of the Quality Engineering (QE) Research Group and head of the Research Cluster on Business Processes and Workflows at QE. Barbara holds a Habilitation degree in Computer Science and Ph.D. from the University of Innsbruck. Barbara's research interests include process model understandability, process of process modeling, integrated process life cycle support, change patterns, process flexibility, user support in flexible process-aware systems, and recommendations to optimize process execution. Barbara has published more than 100 refereed papers, for example, in Data & Knowledge Engineering, Computers in Industry, Science of Computer Programming, Enterprise Information Systems and IET Software and co-author of the book "Enabling Flexibility in Process-aware Information Systems: Challenges, Methods, Technologies" by Springer. Moreover, Barbara is organizer of the successful BPI workshop series.