Expressiveness and Understandability Considerations of Hierarchy in Declarative Business Process Models*

Stefan Zugal[1], Pnina Soffer[2], Jakob Pinggera[1], and Barbara Weber[1]

[1] University of Innsbruck, Austria
{stefan.zugal,jakob.pinggera,barbara.weber}@uibk.ac.at
[2] University of Haifa, Israel
spnina@is.haifa.ac.il

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. In particular, 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 impact on the understandability of hierarchy on a declarative process model. To systematically assess this impact, we present a cognitive-psychology based framework that allows to assess the possible impact of hierarchy on the understandability of the process model.

Key words: Declarative Business Process Models, Hierarchy, Understandability, Cognitive Psychology.

1 Introduction

Using modularization to hierarchically structure information has for decades been identified as a viable approach to deal with complexity [1]. Not surprisingly, business process modeling languages provide support for hierarchical structures, e.g., sub-processes in BPMN and YAWL. However, in general, *"the world does not represent itself to us neatly divided into systems, subsystems... these divisions which we make ourselves"* [2]. 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, sub-processes have been recognized as an important factor influencing model understandability [3, 4], 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 [5] over

* This research is supported by Austrian Science Fund (FWF): P23699-N23 and the BIT fellowship program

5–15 model elements [6] to up to 50 model elements [7]. For declarative process models, which have recently gained attention due to their flexibility [8], the proper usage of modularization is even less clear. While work has been done with respect to the technical support of declarative sub-processes, it remains unclear *whether and when* hierarchy has an influence on the *understandability* of the model. In general, empirical research into the understandability of conceptual models (e.g., ER diagrams or UML statecharts) has shown that hierarchy can have a positive influence [9], negative influence [10] or no influence at all [11]. For declarative process models, the situation is less clear as no empirical studies have been conducted so far. However, as declarative process models appear to be especially challenging to understand [12], it seems particularly important to improve their understandability. 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 into account peculiarities of declarative process models. 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. This, in turn, allows for a systematic empirical investigation in future work.[2]

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

2 Background: Declarative Processes

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 [13]. 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

[2] Please note that even though we take into account declarative models in general, we will make use of the declarative language *ConDec* [13] for the discussion.

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 and termination constraints. *Execution* constraints, on the one hand, restrict the execution of activities, e.g., an activity can be executed at most once. *Termination* constraints, on the other hand, affect the *proper* termination[3] of process instances and specify when process termination is possible. For instance, an activity must be executed at least once before the process can be terminated. Most constraints focus either on execution *or* termination semantics, however, some constraints also combine execution and termination semantics (e.g., the succession constraint [13]).

To illustrate the concept of decarative processes, a model specified in Condec [13] is shown in Fig. 1 a). It contains activities *A* to *F* as well as constraints *C1* and *C2*. *C1* prescribes that A must be executed at least once (i.e., *C1* restricts the termination of process instances). *C2* specifies that *E* can only be executed if *C* has been executed at some point in time before (i.e., *C2* imposes restrictions on the execution of activity E). In Fig. 1 b) an example of a process instance illustrates the semantics of the described constraints. After process instantiation, *A*, *B*, *C*, *D* and *F* can be executed. *E*, however, cannot be executed as *C2* specifies that *C* must have been executed before (cf. grey bar in Fig. 1 b) below "E"). Furthermore, the process instance cannot be terminated as *C1* is not satisfied, i.e., *A* has not been executed at least once (cf. grey area in Fig. 1 b) below "Termination"). The subsequent execution of B does not cause any changes as it is not involved in any constraint. However, after *A* is executed, *C1* is satisfied, i.e., *A* has been executed at least once and thus the process instance can be terminated (cf. Fig. 1 b)—after *e4* the box below "Termination" is white). Then, *C* is executed, satisfying *C2* and consequently allowing *E* to be executed (the box below "E" is white after *e6* occurred). Finally, the execution of *E* does not affect any constraint, thus no changes with respect to constraint satisfaction can be observed. As all termination constraints are still satisfied, the process instance can still be terminated. Please note that declarative process instances have to be terminated explicitly, i.e., the end user must decide when to complete the process instance. Termination constraints thereby specify when termination is allowed, i.e., the process instance I could have been terminated at any point in time after *e4*.

As illustrated in Fig. 1, a process instance can be specified through a list of *events* that describe changes in the life-cycle of *activity instances*, e.g., *"e1: B started"*. In the following, we will denote this list as *execution trace*, e.g., for process instance I: <*e1*, *e2*, *e3*, *e4*, *e5*, *e6*, *e7*, *e8*>. If activities are non-overlapping, we merge subsequent start events and completion events, e.g., <*B started*, *B completed*, *A started*, *A completed*> is abbreviated by <*B*, *A*>.

[3] In the following we will use *termination* as synonym for *proper termination*.

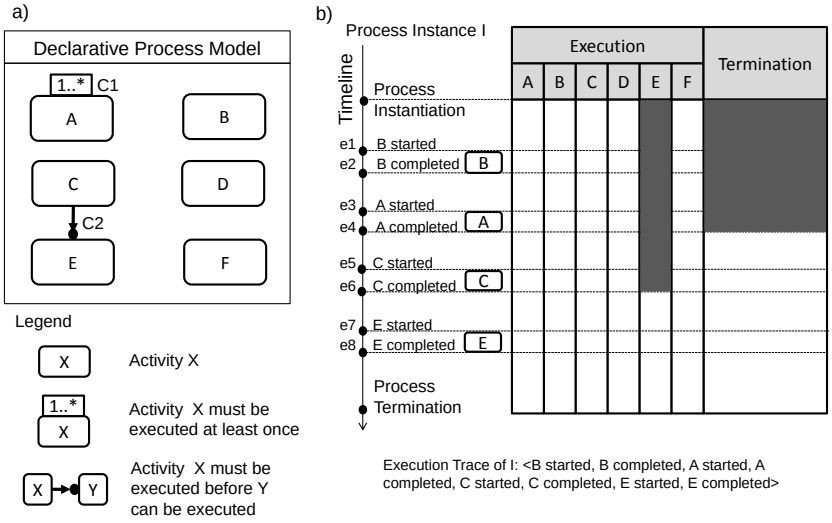

Fig. 1. Executing a declarative process

3 Discussion of Semantics

This section aims at establishing an understanding of the semantics of sub-processes in a declarative model. Based on this, the next section discusses their possible use, and then we turn to discuss their possible effect on model understanding. To our knowledge, the semantics of declarative sub-processes have not been discussed explicitly yet, but their use has been suggested in the context of imperative-declarative model combinations [14]. 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 *individual* 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. [14, 15]). The parent process, however, has no information about the internals of the sub-process, i.e., the sub-process is executed in isolation. Communication with the parent process is done only via the sub-process' life-cycle[4]. Thereby, the life-cycle state of the complex activity reflects the state of the sub-process [14], e.g., when the complex activity is in state *completed*, also the sub-process 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*

[4] We do not take into account communication via input- and output data here, as we focus on control flow behavior only.

neither directly influence the control flow of any parent process, *nor directly influence* the control flow of any sub-process on a layer below.

To illustrate these concepts, consider the process model in Fig. 2 a). It consists of activity A and complex activity B, which in turn contains activities C and D. C and D are connected by a precedence constraint, i.e., D can only be executed if C was executed before. Fig. 2 b) shows an example of a process instance that is executed on this process model. On the left a timeline lists all events that occur during the process execution, e.g., starting or completing an activity. To the right, the enablement of the activities is illustrated. Whenever the area below an activity is colored white, it indicates that this activity is currently enabled. The timeline is to be interpreted the following way: By instantiating the process, activities A and B become enabled, as no constraints restrict their execution. C and D cannot be executed, as they are confined in complex activity B and no instance of B is running yet. Also the subsequent execution of A (*e1*, *e2*) does not change activity enablement. However, with the start of B (*c3*), C becomes enabled, as it can be executed within the new instance of B. Still, D is not enabled yet as the precedence constraint is not satisfied. After C is executed (*e4*, *e5*), the precedence constraint is satisfied, therefore also D becomes enabled. After the execution of D (*e6*, *e7*), the user decides to complete sub-process B (*e8*). Hence, C and D cannot be executed anymore. Still, A and B are enabled as they can be executed directly within process instance I. Finally, after the process instance is completed by the end user through explicit termination, no activity is enabled anymore.

4 Using Hierarchy in Declarative Models

Regardless of the modeling language, hierarchy allows to structure models and to hide modeling elements in sub-models. In this section, the use of hierarchy, given the semantics of Section 3, is discussed.

4.1 Running Example

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. Due to space restrictions, the examples are not described in detail, but will be used in the following for illustration purposes.

4.2 Preconditions for Using Sub-Processes

While for imperative models, any Single-Entry-Single-Exit fragment can be extracted to a sub-process [16, 17], in declarative models the structure is not informative enough. Rather, two main conditions should hold for the introduction

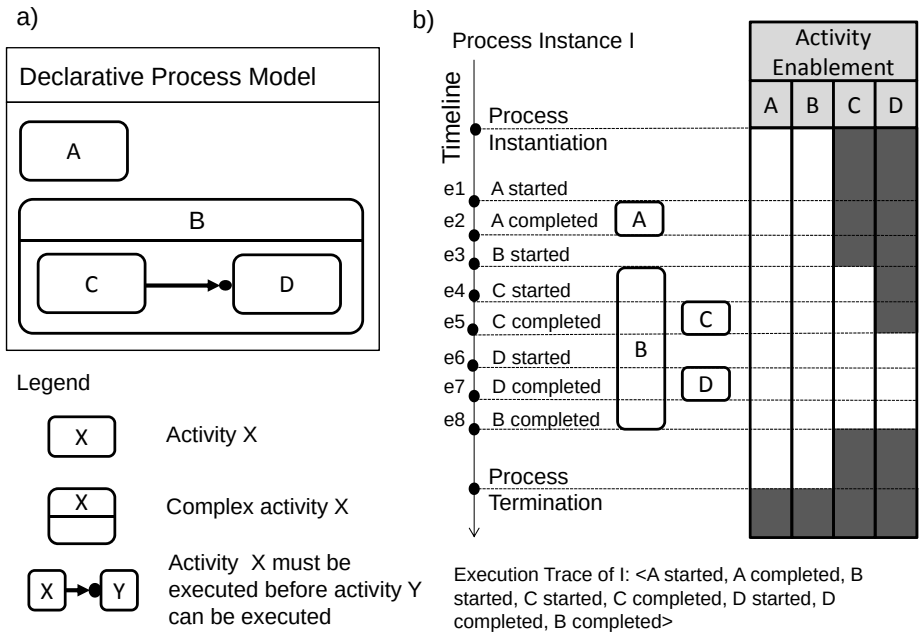

Fig. 2. Execution of a sub-process

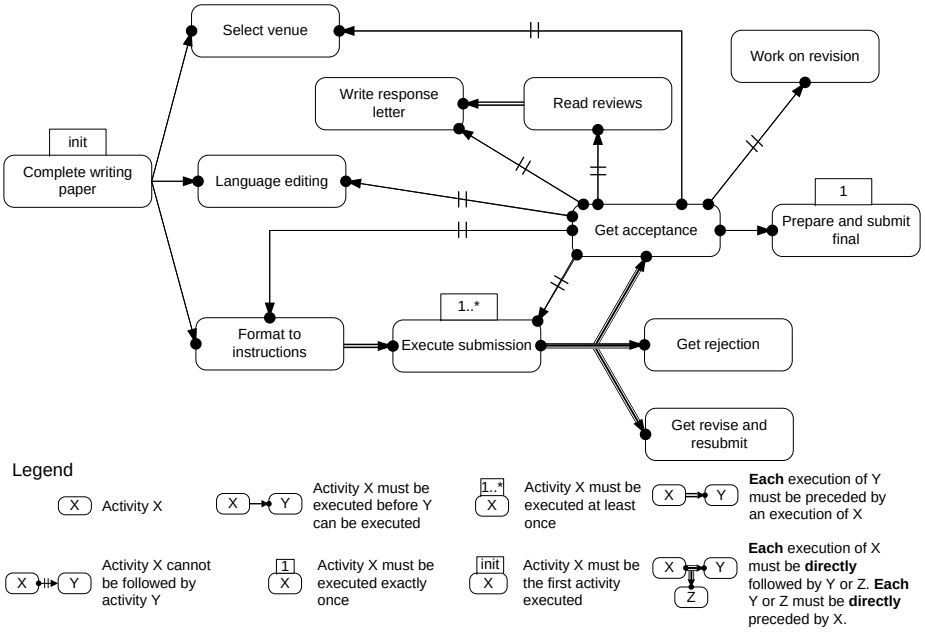

Fig. 3. Example of a flat model

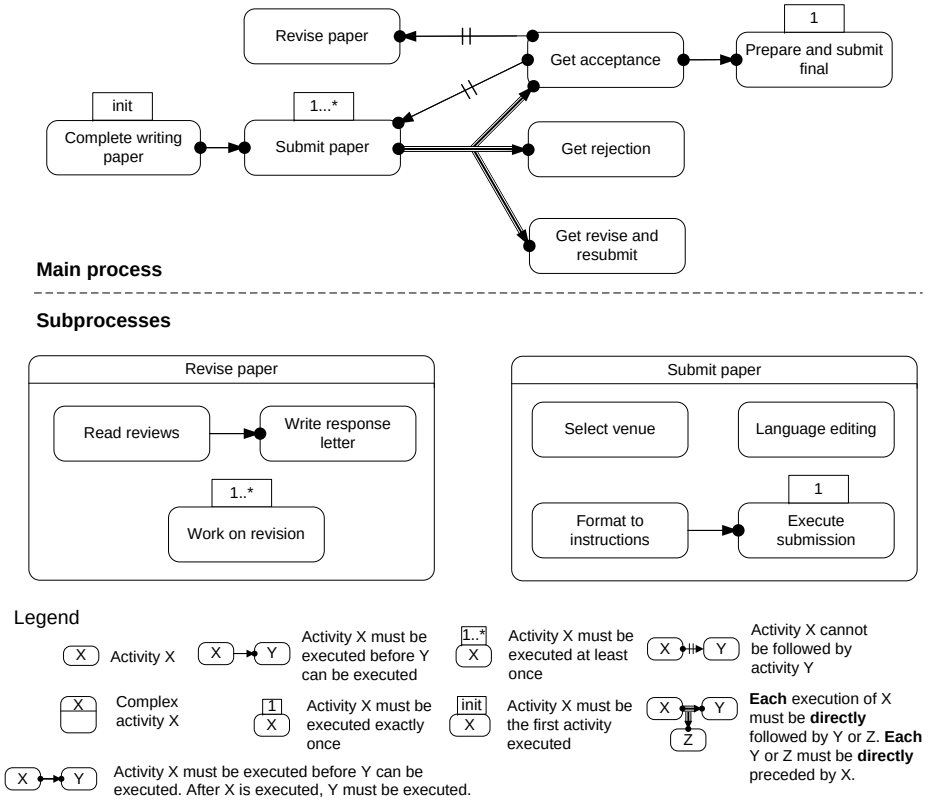

Fig. 4. Example of a hierarchical model

of sub-processes. First, the activities in a sub-process should relate to a certain intention [18] to be fulfilled. For instance, in Fig. 4, *Read reviews*, *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 executed in isolation from the top-level process. This is due to the local nature of the constraints within the sub-process, and the lack of communication with the parent process, as discussed in Section 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, *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 so a single constraint, we refer this to as *aggregation of constraints*.

4.3 Enhanced Expressiveness

For *imperative* process models, hierarchical decomposition is viewed as a structural measure that may impact model understandability [19], 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 Section 4.2). 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 model M in Fig. 5, which solely uses constraints defined in [20]. The chained precedence constraint between C and D specifies that for each execution of D, sub-process C has to be executed directly before. When executing sub-process 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*. 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 C*. Such behavior cannot be modeled without hierarchy, using the same set of constraints.

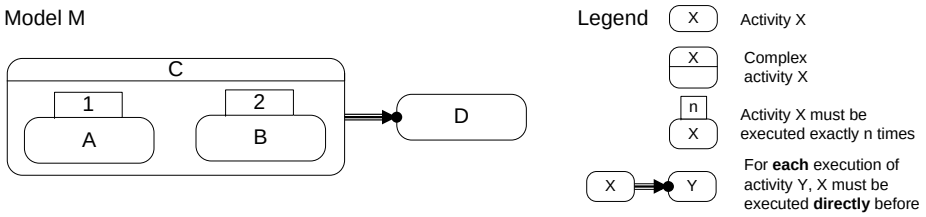

Fig. 5. Enhanced expressiveness

4.4 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 [16], 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, only those hierarchical models that do not make use of the enhanced expressiveness can 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.

5.1 Framework for Assessing Understandability

The influence of hierarchy on model understandability has been investigated in a number of different modeling languages, such as ER-Models [21], imperative business process models [9] and UML Statecharts [11] (for an overview see [19]). While reported results do not entirely clarify when and how understandability is affected, a trade-off between (sub)model size and degree of hierarchy can be observed. For instance, in small models hierarchy may have no [11] or even a negative impact [10], while for large models a positive influence could be observed [9].

In [19], 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 [22]). In this work we present an enhanced version that is still generic but also takes into account the idiosyncrasies of hierarchy in declarative process models. The central concept of the framework is *mental effort* [23], 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 reader to *switch attention between fragments* and *integrating* information from fragments.

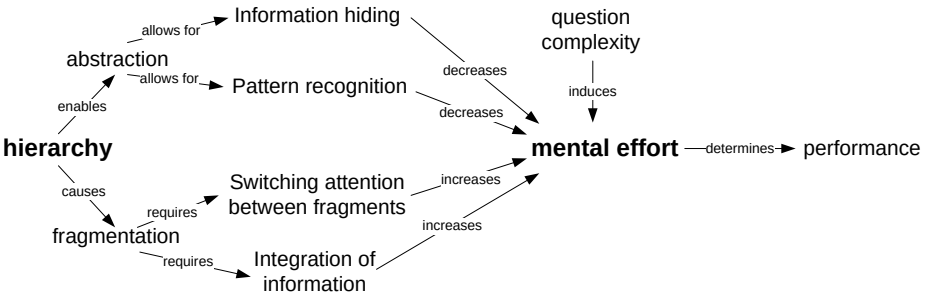

Fig. 6. Framework for assessing hierarchy, adapted from [19]

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 reader, leading to decreased mental effort, as argued in [21]. From the perspective of cognitive psychology, this phenomenon can be explained by the concept of *attention management* [24]. 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 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 [24, 25], 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 Fig. 3 and Fig. 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 reader first has to mentally group together activities before the overall intention of the process becomes clear.

Fragmentation. Empirical evidence shows that the influence of hierarchy can range from positive over neutral to negative (cf. [11, 10, 26, 21, 9]). 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 reader has to *switch attention* between the parent model and the sub-process. In addition, the reader has to *mentally integrate* the sub-process into the parent model, i.e., interpreting constraints in the context of the parent process. From the perspective of cognitive psychology, these phenomena are known to increase mental effort and referred to as *split-attention effect* [27]. To exemplify this effect, consider the process model in Fig. 4. 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 reader has to split attention between these two activities. In addition, the reader 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*.

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. [28, 10, 26]).

5.2 Impact of Idiosyncrasies on Understandability

In Section 4, we have shown that hierarchy enhances expressiveness and allows to aggregate constraints. In the following, we will discuss the impact of these two phenomena on understandability.

Enhanced Expressiveness and Complex Mental Integration. As argued, hierarchy provides enhanced expressiveness. However, this also comes at a price, as the constraint that is referring to a sub-process has to be integrated with the

semantics of the constraints *within* the sub-process. To illustrate such integrations, consider the process model in Fig. 4. Activity *Work on revision* has to be executed at least once, i.e., is mandatory. However, this activity is contained in complex activity *Revise paper*, which is optional. In other words, *Work on revision* is mandatory for *Revise paper*, which is optional for the main process. Consequently, also *Work on revision* is optional for the main process.

Such mental integrations can be found in any hierarchical conceptual model. For instance, in an imperative process model, mental integration refers to transferring the token from the parent process to the start event of the sub-process. As argued, however, integrations are particularly complex in declarative process models. Hence, it can be expected that a strong influence on the understandability can be observed.

Aggregation of Constraints. As discussed in Section 4.3, hierarchy allows to aggregate and thus reduce the number of constraints. In the context of the proposed framework, we can identify three forces. Positively, aggregation reduces the number of constraints, hence hiding information. In addition, a reduced number of constraints fosters the the layout of the process model. This, in turn, supports the recognition of patterns, i.e., making the model easier to understand. Negatively, complex mental integration operations, as discussed before, may diminish the described gains. Whether positive or negative influences predominate will have to be investigated empirically, as discussed in the following.

5.3 Discussion

So far we argued that hierarchy in declarative process models can be attributed to increases as well as decreases in understandability. In the following, we will discuss the impact of the identified influences. Positively, we see a big potential for hierarchy in declarative process models. In an imperative process model, control flow is modeled explicitly. Hence, process models are usually structured according to their control flow. Such a strategy is in general not possible for a declarative process model, as constraints do not necessarily prescribe sequential information. Sub-processes, however, allow to group activities and thereby to introduce structure to the model. Sub-processes, however, allow to group activities by a mutual intention they serve and thereby to introduce structure to the model and add higher-level information. As argued in our framework, this allows *recognizing patterns* and makes it easier to grasp the intention of a business process (cf. Fig. 3 and Fig. 4). Also the ability of sub-processes to *hide information*, i.e., activities and constraints, can be expected to contribute to the understandability of models. It is assumed that several interconnected constraints quickly become challenging for the human mind [13, 12, 29]. Hence, hiding information can be expected to be especially beneficial in declarative process models.

On the other hand, as argued in Section 5.2, we assume that the integration of constraints poses a significant challenge for the reader. In particular, it is not clear yet whether an average process modeler is able to efficiently perform such mental integrations. This is, however, necessary for the meaningful application

of enhanced expressiveness by hierarchy. If efficient mental integration was not possible, enhanced expressiveness would be rendered useless as resulting models would be hardly understandable.

The presented framework can be seen as a first step towards a systematic assessment of the impact of hierarchy on understandability in declarative process models. Even though it is based upon well-established concepts from cognitive psychology, the claims still have to be empirically challenged. In particular, we postulated that the integration of constraints poses a significant, but manageable challenge for the reader. Similarly, we assume that *large* declarative process models tend to be too complex for humans to deal with (cf. [13]). To corroborate the postulated claims, we are currently planning a thorough empirical investigation, cf. Section 7.

6 Related Work

In this work we discussed characteristics of hierarchy in declarative process models and the impact on understandability. The impact of hierarchy on understandability has been studied in various conceptual modeling languages, such as imperative business process models [30, 9], ER diagrams [21, 31] and UML statechart diagrams [10, 32, 28] (an overview is presented in [19]). Still, none of these works deals with the impact of hierarchy on understandability in declarative process models. The understandability of declarative process models in general has been investigated in the work of Zugal et al. [12, 33, 34], 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 [7] modeling guidelines are presented that target to improve the understandability of imperative process models. The understandability of imperative process models is investigated empirically in [35, 36]. Finally, in [20, 13] the technical aspects of declarative business process models, such as the definition of modeling languages or verification of models is investigated. In contrast to this work, understandability aspects are neglected and the unique semantics and expressiveness enabled by sub-processes is not elaborated.

7 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 presumably poses a significant challenge for the reader. In addition, switching between sub-processes may compromise the understandability of respective models. We acknowledge that, even though the framework is based on well-established concepts from cognitive psychology, an empirical validation still has to be conducted.

In this sense, our next steps clearly focus on empirical validation. In particular, two main research directions are envisioned. First, we will investigate whether information hiding lowers cognitive load and hence improves understandability. Second, the integration of constraints and the required mental effort will be scrutinized.

References

1. Parnas, D.L.: On the Criteria to be Used in Decomposing Systems into Modules. Communications of the ACM **15** (1972) 1053–1058
2. Goguen, J.A., Varela, F.J.: Systems and Distinctions; Duality and Complementarity. Int. J. Gen. Syst. **5** (1979) 31–43
3. Davies, R.: Business Process Modelling With Aris: A Practical Guide. Springer (2001)
4. Damij, N.: Business process modelling using diagrammatic and tabular techniques. Business Process Management Journal **13** (2007) 70–90
5. Sharp, A., McDermott, P.: Workow Modeling: Tools for Process Improvement and Application Development. Artech House (2011)
6. Kock, N.F.: Product flow, breadth and complexity of business processes: An empirical study of 15 business processes in three organizations. Business Process Re-engineering & Management Journal **2** (1996) 8–22
7. Mendling, J., Reijers, H.A., van der Aalst, W.M.P.: Seven process modeling guidelines (7pmg). Information & Software Technology **52** (2010) 127–136
8. Pesic, M., Schonenberg, H., Sidorova, N., van der Aalst, W.: Constraint-Based Workflow Models: Change Made Easy. In: Proc. CoopIS '07. (2007) 77–94
9. Reijers, H., Mendling, J., Dijkman, R.: Human and automatic modularizations of process models to enhance their comprehension. Inf. Systems **36** (2011) 881–897
10. Cruz-Lemus, J., Genero, M., Piattini, M.: Using controlled experiments for validating uml statechart diagrams measures. In: Software Process and Product Measurement. Volume 4895 of LNCS. Springer Berlin / Heidelberg (2008) 129–138
11. Cruz-Lemus, J., Genero, M., Piattini, M., Toval, A.: Investigating the nesting level of composite states in uml statechart diagrams. In: Proc. QAOOSE '05. (2005) 97–108
12. Zugal, S., Pinggera, J., Weber, B.: Toward Enhanced Life-Cycle Support for Declarative Processes. JSME (2011) DOI: 10.1002/smr.554.
13. Pesic, M.: Constraint-Based Workflow Management Systems: Shifting Control to Users. PhD thesis, TU Eindhoven (2008)
14. Pesic, M., Schonenberg, H., van der Aalst, W.: DECLARE: Full Support for Loosely-Structured Processes. In: Proc. EDOC '07. (2007) 287–298
15. OMG: BPMN Version 2.0. http://www.omg.org/spec/BPMN/2.0/PDF/ (2011)

16. Weber, B., Reichert, M., Mendling, J., Reijers, H.A.: Refactoring large process model repositories. Computers in Industry **62** (2011) 467–486
17. Weber, B., Reichert, M., Rinderle, S.: Change Patterns and Change Support Features - Enhancing Flexibility in Process-Aware Information Systems. DKE **66** (2008) 438–466
18. Soffer, P., Rolland, C.: Combining Intention-Oriented and State-Based Process Modeling. In: Proc. ER '05. (2005) 47–62
19. Zugal, S., Pinggera, J., Weber, B., Mendling, J., Reijers, H.A.: Assessing the Impact of Hierarchy on Model Understandability—A Cognitive Perspective. In: Proc. EESSMod '11. (2011) 18–27
20. Montali, M., Pesic, M., van der Aalst, W., Chesani, F., Mello, P., Storari, S.: Declarative Specification and Verification of Service Choreographies. ACM Trans. Web **4** (2010) 1–62
21. Moody, D.L.: Cognitive Load Effects on End User Understanding of Conceptual Models: An Experimental Analysis. In: Proc. ADBIS '04. (2004) 129–143
22. Zugal, S., Pinggera, J., Weber, B.: Assessing process models with cognitive psychology. In: Proc. EMISA '11. (2011) 177–182
23. Sweller, J.: Cognitive load during problem solving: Effects on learning. Cognitive Science **12** (1988) 257–285
24. Larkin, J.H., Simon, H.A.: Why a Diagram is (Sometimes) Worth Ten Thousand Words. Cognitive Science **11** (1987) 65–100
25. Scaife, M., Rogers, Y.: External cognition: how do graphical representations work? Int.J. Human-Computer Studies **45** (1996) 185–213
26. 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: Proc. ER Workshops. (2005) 12–22
27. Sweller, J., Chandler, P.: Why Some Material Is Difficult to Learn. Cognition and Instruction **12** (1994) 185–233
28. 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 Software Eng **25** (2009) 685–719
29. Weber, B., Reijers, H.A., Zugal, S., Wild, W.: The Declarative Approach to Business Process Execution: An Empirical Test. In: Proc. CAiSE '09. (2009) 270–285
30. Reijers, H., Mendling, J.: Modularity in Process Models: Review and Effects. In: Proc. BPM '08. (2008) 20–35
31. Shoval, P., Danoch, R., Balabam, M.: Hierarchical entity-relationship diagrams: the model, method of creation and experimental evaluation. Requirements Engineering **9** (2004) 217–228
32. Cruz-Lemus, J.A., Genero, M., Morasca, S., Piattini, M.: Using Practitioners for Assessing the Understandability of UML Statechart Diagrams with Composite States. In: Proc. ER Workshops '07. (2007) 213–222
33. Zugal, S., Pinggera, J., Weber, B.: The impact of testcases on the maintainability of declarative process models. In: Proc. BPMDS '11. (2011) 163–177
34. Zugal, S., Pinggera, J., Weber, B.: Creating Declarative Process Models Using Test Driven Modeling Suite. In: Proc. CAiSE Forum '11. (2011) 1–8
35. Reijers, H.A., Mendling, J.: A Study into the Factors that Influence the Understandability of Business Process Models. SMCA **41** (2011) 449–462
36. Mendling, J., Reijers, H.A., Cardoso, J.: What Makes Process Models Understandable? In: Proc. BPM '07. (2007) 48–63