Process Change Patterns: Recent Research, Use Cases, Research Directions

Manfred Reichert and Barbara Weber

Abstract In previous work, we introduced change patterns to foster a systematic comparison of process-aware information systems with respect to change support. This paper revisits change patterns and shows how our research activities have evolved. Further, it presents characteristic use cases and gives insights into current research directions.

1 Introduction

Information systems (IS) are increasingly aligned in a process-oriented way. This emerging generation of IS is referred to as process-aware information systems (PAIS) [1]. A PAIS should support real-world processes properly, i.e., there should be no mismatch between the processes implemented by it and those existing in reality. Hence, advanced support is needed for customizing a PAIS to its application environment as well as for quickly adapting implemented processes to changing needs. The increasing demand for process change support poses new challenges for IS engineers and requires the use of change enabling technologies.

Accordingly, a method is required that allows PAIS engineers to systematically assess the change capabilities of available technologies. In [2], we introduced change patterns as well as change support features to enable such a systematic assessment of PAIS with respect to process change support. In particular, change patterns allow for high-level process adaptations. In turn, change support features

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

B. Weber
University of Innsbruck, Innsbruck, Austria
e-mail: barbara.weber@uibk.ac.at

summarize fundamental features to be provided by a PAIS in order to change and evolve implemented processes in a correct, robust and secure way.

This paper discusses how our research on change patterns has evolved, how they have been used in theory and practice, and what research directions are.

2 Background: Process Change Patterns

Originally, in [2] we introduced 17 patterns for realizing control flow changes. These patterns reduce the complexity of process changes and raise the level for expressing changes by providing abstractions above the level of primitive change operations. To structure the change patterns, we divided them into *adaptation patterns* and *change patterns for predefined changes* (cf. Fig. 2 in [2]). While the former enable structural changes of a process schema, the latter allow process participants to add information regarding unspecified parts of a process schema during run-time.

An *adaptation pattern (AP)* enables structural changes of process schemes. AP1 (AP2) allows inserting (deleting) a process fragment. Moving and replacing fragments is supported by AP3 (Move Process Fragment), AP4 (Replace Process Fragment), AP5 (Swap Process Fragment), and AP14 (Copy Process Fragment). AP6 and AP7 allow adding or removing levels of hierarchy: the extraction of a sub-process from a process schema is supported by AP6, whereas the inclusion of a sub-process into a process schema is supported by AP7. Patterns AP8–AP12 support adaptations of control dependencies: embedding a process fragment in a loop (AP8), parallelizing a process fragment (AP9), embedding a process fragment in a conditional branch (AP10), and adding/removing control dependencies (AP11, AP12). Finally, AP13 allows changing transition conditions. Generally, the region to which an adaptation pattern is applied may be chosen dynamically. Hence, adaptation patterns are well suited for realizing ad-hoc changes and coping with the evolving nature of business processes [1]. For each adaptation pattern, we have provided a name, a description, an illustrating example, a description of the problem it addresses, a couple of design choices, remarks regarding its implementation, and references to related patterns. In this context, design choices allow parameterizing change patterns keeping the number of distinct patterns manageable.

Patterns for changes in pre-defined regions allow for better dealing with uncertainty by deferring decisions regarding the exact control-flow to the run-time. Instead of requiring a process model to be fully specified prior to execution, parts of the model may remain unspecified. As opposed to adaptation patterns, whose application is not restricted a priori to a particular process part, the parts of a process schema that may be changed or expanded are constrained. In this category, we identified four patterns: Late Selection (PP1), Late Modeling (PP2) and Late Composition of Process Fragments (PP3), and Multi-Instance Activity (PP4). These four patterns differ regarding the parts that may remain unspecified resulting in a different degree of freedom during run-time.

3 Recent Research on Process Change Patterns

In recent work we have detailed the change patterns and provided empirical evidence for them. Further, we have formalized and implemented them. In detail:

Detailing change patterns and empirical evidence. We extended our original work in [3], which provides an in-depth description of all change patterns; it describes the pattern selection criteria, the data sources used, and the procedure applied for pattern identification. Further, it discusses how the patterns were identified based on the analysis of large process model collections from the healthcare and automotive domains. Finally, Weber et al. [3] introduces additional patterns and provides an extended pattern-based evaluation of selected approaches from industry as well as academia.

Change pattern formalization. To obtain unambiguous pattern descriptions and ground pattern implementation as well as pattern-based analyses on a sound basis, we provided a formal semantics for change patterns in [4]. For each change pattern, its formal semantics is specified by comparing the execution traces producible on a process schema before and after applying the pattern to it. The formalization is independent from any process meta model and thus allows implementing the patterns in a variety of process support tools.

Pattern implementation. The change patterns were implemented in an adaptive PAIS – the AristaFlow BPM Suite [5]. The adaptation patterns are realized in terms of high-level change operations, which can be used for creating and changing process schemes. Hence, flexible exception handling and controlled process evolution become possible. Further, adaptation patterns are associated with pre-/post-conditions to ensure structural and behavioral soundness of a process schema after pattern application; i.e., correctness by construction is ensured.

Recently, we complemented the existing workflow patterns by a set of time patterns to make PAIS comparable with respect to their ability to deal with temporal constraints [6].

4 Characteristic Use Cases for Change Patterns

On one hand, change patterns provide the basis for realizing changes in different stages of the process life cycle [7]. On the other, they serve as benchmark for evaluating change support in existing languages and tools.

4.1 Supporting Process Changes Along the Process Life Cycle

We first discuss fundamental use cases for realizing changes in different stages of the process life cycle:

Process schema creation. Change patterns have been used for intelligent process schema creation [8]. For example, AristaFlow allows modeling a sound process schema based on an extensible set of adaptation patterns [5]. Only those patterns may be applied in a given context, which do not violate the soundness of the process schema. In turn, Gschwind et al. [9] describes a set of pattern compounds, similar to the adaptation patterns, allowing for the context-sensitive selection and composition of workflow patterns during process modeling. Finally, adaptation patterns have been used for the model-based integration of services into business applications at later stages during the process life cycle [10].

Process schema configuration. The configuration of a reference process schema constitutes another use case for adaptation patterns. Provop, for example, allows creating a process variant by applying a sequence of adaptation patterns (e.g., AP1, AP2 or AP3) to the given reference schema [11]. By utilizing the semantics of the adaptation patterns applied in a given configuration setting, it is further ensured that the resulting process variant schema is sound [12].

Process instance change. An important use case is to enable actors to deviate from a pre-specified process schema at run-time, e.g., to cope with exceptions. For this purpose, AristaFlow supports instance-specific changes based on the same adaptation patterns as used for process modeling [5]. Further, it utilizes the semantics of the applied adaptation patterns to ensure correctness of the resulting process instance schema. Recently, approaches aiming at automated instance changes have emerged. Usually, they only consider a subset of the adaptation patterns. For example, Q-Advice uses AP1 and its variants to automatically inject quality measure activities into the workflows of software engineers at run-time. The activities to be added are determined situationally using contextual knowledge and quality goal tracking [13]. A more generic approach to automate instance adaptations, which is based on declarative processes and planning, is described in [14]. Regarding ad-hoc changes, Kumar et al. [15] additionally ensures compliance of process instance adaptations with defined semantic constraints. For this, integer programming formulation is used to validate the applied adaptation patterns against the given set of semantic constraints (AP1–AP5 are considered). An approach for the flexible support of product development processes is presented in [16]; the sub-processes of such a process, which refine analysis, synthesis, and verification activities, may be dynamically selected to allow for a flexible process execution without need for structural adaptations. Thereby, a subset of the patterns for changes in pre-defined regions is considered (i.e., PP1–PP3).

Process schema evolution. Adaptive PAIS allow for schema evolution considering version management and the migration of already running process instances to the new schema version. Gerth et al. [17] presents techniques for detecting and resolving conflicting change operations, which rely on selected adaptation patterns and their semantics. In turn, Küster et al. [18] shows how to compute a sequence of adaptation patterns required to transform a given schema version into another one.

Both scenarios consider AP1, AP2, and AP3. Particularly, adaptation patterns play a crucial role for ensuring the correctness of schema changes and instance migrations. In AristaFlow, schema evolution is based on the same adaptation patterns as used for process modeling and ad-hoc changes [5]. Thereby, pattern semantics is utilized to cope with conflicting changes at the type and instance level, to increase the number of migratable process instances, and to efficiently represent applied changes [19–21]. Note that similar concepts exist for evolving service compositions [22]. Furthermore, continuous process improvement, relying on case-based reasoning and adaptation patterns, is considered in [23]. Finally, Jamshidi and Pahl [24] introduces patterns for co-evolving processes and software architectures. These patterns are based on selected adaptation patterns and allow describing the impact a business process change has on corresponding software architectures.

Process schema refactoring. A specific kind of schema evolution is provided by process schema refactorings; i.e., syntactical transformations of a process schema not changing its behavior. Examples of such refactorings and their relation to adaptation patterns (e.g., AP6 and AP7) are discussed in [25].

Process change reuse. When handling exceptions, it might be useful to reuse changes applied in similar problem contexts earlier [26]. For example, ProCycle fosters change reuse based on case-based reasoning and semantic change annotations [7]. Further, it supports AP1–AP5 and utilizes their specific semantics to adjust parameter settings of recorded changes when reusing them.

Process schema comparison. Comparing two process schemes is crucial to decide how similar the schemes are or how to derive the one from the other. In this context, adaptation patterns can be used to describe the structural difference (i.e., edit distance) between schemes in terms of high-level changes. Based on specific variants of patterns AP1–AP3, Li et al. [27] presents a technique that allows determining this difference. A similar approach is presented in [28].

Process change analysis. Adaptive PAIS capture process changes in *change logs*, which record applied adaptation patterns and their parameter settings. For change analysis, different techniques exist. Based on AP1–AP3, Günther et al. [29] applies process mining to change logs to discover *change processes* providing an aggregated visualization of all changes. In turn, MinAdept does not presume the existence of a change log, but allows analyzing a collection of process variants derived from the same schema [30]; algorithms are provided discovering a reference process schema whose average edit distance to the process variants is minimal.

In summary, process change patterns are relevant for a variety of use cases in the process life cycle. As shown, the patterns have served as basis for the design and implementation of techniques supporting these use cases. While tools like AristaFlow enable a broad support of most use cases and adaptation patterns, other proposals only consider a specific use case and a subset of the adaptation patterns.

4.2 Assessing and Designing Process Change Frameworks

Change patterns have been used for realizing pattern catalogs for specific modeling languages, assessing existing PAIS, and enabling user-friendly changes. Examples of corresponding approaches are presented in the following.

Realizing a pattern catalog for a specific modeling language. Döhring et al. [31] combines change, exception and time patterns into a BPMN pattern catalog. Change patterns are referred to as generic patterns, which are tailored and extended to be applicable to BPMN. In turn, Tragatschnig and Zdun [32] uses the adaptation patterns for designing a pattern catalog for BPEL schema changes.

Assessing existing approaches. A measure for a pattern-based assessment of service orchestration languages is defined in [33]. In particular, the designed pattern catalog includes the patterns for changes in predefined regions (i.e., PP1–PP4) and discusses how they are supported in existing BPEL dialects.

Enabling user-friendly changes. Kolb et al. [34] presents an approach enabling end users to change large process schemes based on personalized process views; AP1, AP2, and AP8–AP10 may be applied to a process view, followed by the propagation of the defined changes to the underlying process schema. In turn, Kolb et al. [35] introduces a user-centric approach for creating, changing and visualizing process schemes based on the Concurrent Task Tree (CTT) – a task modeling language known from end-user programming. Thereby, the described adaptation patterns are mapped to CTT change operations. Finally, Kolb et al. [36] presents a controlled experiment that investigates the way users create and change process schemes on multi-touch devices. Based on this, a gesture set for realizing adaptation patterns AP1, AP2, AP6, AP7, AP8, AP10, and AP11 on multi-touch devices is designed.

5 Research Directions

When using change patterns for modeling, the quality of process schemes might increase. Particularly appealing in this context is the mentioned correctness by construction. However, the use of change patterns implies a different way of creating process schemes compared to the use of change primitives. First of all, the correctness-by-construction principle imposes a rather structured way of modeling and hence constraints on change pattern combinations. In addition, the exact set of change patterns (e.g., presence vs. non-presence of the move pattern) might determine how patterns have to be combined to create a process fragment. While the creation of process schemes based on change primitives has caused attention in recent years [37], only little is known about the process of process modeling when utilizing change patterns. To obtain an in-depth understanding of it, we are currently working on empirical studies on the use of change patterns.

References

1. Reichert, M., Weber, B.: Enabling Flexibility in Process-Aware Information Systems - Challenges, Methods, Technologies. Springer (2012)
2. Weber, B., Rinderle, S., Reichert, M.: Change patterns and change support features in process-aware information systems. In: Proc. CAiSE'07. (2007) 574–588
3. Weber, B., Reichert, M., Rinderle-Ma, S.: Change patterns and change support features - enhancing flexibility in process-aware information systems. Data and Knoweldge Engineering **66** (2008) 438–466
4. Rinderle-Ma, S., Reichert, M., Weber, B.: On the formal semantics of change patterns in process-aware information systems. In: Proc. ER'08. (2008) 279–293
5. Dadam, P., Reichert, M.: The ADEPT project: a decade of research and development for robust and flexible process support. Comp Scie - R&D **23** (2009) 81–97
6. Lanz, A., Weber, B., Reichert, M.: Time patterns for process-aware information systems. Requirements Engineering (2013)
7. Weber, B., Reichert, M., Wild, W., Rinderle-Ma, S.: Providing integrated life cycle support in process-aware information systems. Int'l Journal of Cooperative Information Systems **18** (2009) 115–165
8. Smirnov, S., Weidlich, M., Mendling, J., Weske, M.: Object-sensitive action patterns in process model repositories. In: Proc. BPM'10 Workshops. (2010) 251–263
9. Gschwind, T., Koehler, J., Wong, J.: Applying patterns during business process modeling. In: Proc BPM'08. (2008) 4–19
10. Heller, M., Allgaier, M.: Model-based service integration for extensible enterprise systems with adaptation patterns. In: ICE-B. (2010) 163–168
11. Hallerbach, A., Bauer, T., Reichert, M.: Capturing variability in business process models: The Provop approach. Journal of Software Maintenance and Evolution: Research and Practice **22** (2010) 519–546
12. Hallerbach, A., Bauer, T., Reichert, M.: Guaranteeing soundness of configurable process variants in Provop. In: Proc. CEC'09. (2009) 98–105
13. Grambow, G., Oberhauser, R., Reichert, M.: Contextual injection of quality measures into software engineering processes. Int'l J Adv in Software **4** (2011) 76–99
14. Marrella, A., Mecella, M., Russo, A.: Featuring automatic adaptivity through workflow enactment and planning. In: Proc CollaborateCom'11. (2011) 372–381
15. Kumar, A., Yao, W., Chu, C.H., Li, Z.: Ensuring compliance with semantic constraints in process adaptation with rule-based event processing. In: Proc RuleML'10. (2010) 50–65
16. Reichel, T., Rünger, G., Steger, D.: Flexible workflows for an energy-oriented product development process. In: Proc ISSS/BPSC'10. (2010) 243–254
17. Gerth, C., Küster, J., Luckey, M., Engels, G.: Detection and resolution of conflicting change operations in version management of process models. SOSYM (2011) 1–19
18. Küster, J., Gerth, C., Engels, G.: Dynamic computation of change operations in version management of business process models. In: ECMFA'10. (2010) 201–216
19. Rinderle, S., Reichert, M., Dadam, P.: On dealing with structural conflicts between process type and instance changes. In: Proc. BPM'04, Potsdam (2004) 274–289
20. Rinderle-Ma, S., Reichert, M., Weber, B.: Relaxed compliance notions in adaptive process management systems. In: Proc. ER'08. (2008) 232–247
21. Rinderle, S., Reichert, M., Jurisch, M., Kreher, U.: On representing, purging, and utilizing change logs in process management systems. In: BPM'06. (2006) 241–256
22. Andrikopoulos, V., Benbernou, S., Papazoglou, M.P.: Managing the evolution of service specifications. In: Proc. CAiSE'08. (2008) 359–374
23. Kim, D., Lee, N., Kang, S.H.: An approach to continuous process improvement based on case-based reasoning and process change patterns. IJICIC **8** (2011)
24. Jamshidi, P., Pahl, C.: Business process and software architecture model co-evolution patterns. In: Proc. MISE'12. (2012) 91–97

25. Weber, B., Reichert, M., Mendling, J., Reijers, H.A.: Refactoring large process model repositories. Computers in Industry **62** (2011) 467–486
26. Aghakasiri, Z., Mirian-Hosseinabadi, S.H.: Workflow change patterns: Opportunities for extension and reuse. In: Proc. SERA'09. (2009) 265–275
27. Li, C., Reichert, M., Wombacher, A.: On measuring process model similarity based on high-level change operations. In: Proc. ER'08. (2008) 248–264
28. Küster, J.M., Gerth, C., Förster, A., Engels, G.: Detecting and resolving process model differences in the absence of a change log. In: BPM'08. (2008) 244–260
29. Günther, C.W., Rinderle, S., Reichert, M., van der Aalst, W.M.P.: Change mining in adaptive process management systems. In: Proc. CoopIS'06. (2006) 309–326
30. Li, C., Reichert, M., Wombacher, A.: Mining business process variants: Challenges, scenarios, algorithms. Data & Knowledge Engineering **70** (2011) 409–434
31. Döhring, M., Zimmermann, B., Karg, L.: Flexible workflows at design- and runtime using bpmn2 adaptation patterns. In: Proc. BIS. (2011) 25–36
32. Tragatschnig, S., Zdun, U.: Runtime process adaptation for bpel process execution engines. In: EDOCW, IEEE Computer Society (2011) 155–163
33. Lenhard, J., Schönberger, A., Wirtz, G.: Edit distance-based pattern support assessment of orchestration languages. In: OTM Conferences (1). (2011) 137–154
34. Kolb, J., Kammerer, K., Reichert, M.: Updatable process views for user-centered adaption of large process models. In: Proc. ICSOC'12. (2012) 484–498
35. Kolb, J., Reichert, M., Weber, B.: Using concurrent task trees for stakeholder-centered modeling and visualization of business processes. In: S-BPM ONE. (2012)
36. Kolb, J., Rudner, B., Reichert, M.: Towards gesture-based process modeling on multi-touch devices. In: Proc. CAiSE Workshops. (2012) 280–293
37. Pinggera, J., et al: Modeling styles in business process modeling. In: BMMDS/EMMSAD. (2012) 151–166