Towards Run-time Flexibility for Process Families: Open Issues and Research Challenges*

Clara Ayora[1], Victoria Torres[1], Manfred Reichert[2], Barbara Weber[3], and Vicente Pelechano[1]

[1] Universitat Politècnica de València
{cayora,vtorres,pele}@pros.upv.es
[2] University of Ulm, Germany
manfred.reichert@uni-ulm.de
[3] University of Innsbruck, Austria
barbara.weber@uibk.ac.at

Abstract. The increasing adoption of process-aware information systems and the high variability of business processes in practice have resulted in process model repositories with large collections of related process variants (i.e., process families). Existing approaches for variability management focus on the modeling and configuration of process variants. However, case studies have shown that run-time configuration and re-configuration as well as the evolution of process variants are essential as well. Effectively handling process variants in these lifecycle phases requires deferring certain configuration decisions to the run-time, dynamically re-configuring process variants in response to contextual changes, adapting process variants to emerging needs, and evolving process families over time. In this paper, we characterize these flexibility needs for process families, discuss fundamental challenges to be tackled, and provide an overview of existing proposals made in this context.

Key words: Run-time Flexibility, Variability, Process Families

1 Introduction

In recent years, the increasing adoption of *Process-aware Information Systems* (PAISs) has resulted in large process model repositories [4]. Since *Business Process* (BP) models often vary, depending on their application context [6, 19], these repositories usually comprise large collections of related *process model variants* (*process variants* for short) [15]. Such process variants pursue the same or similar business objective (e.g., treatment of a patient or maintenance of vehicles in a garage), but may differ in their logic (i.e., process logic) due to varying application context at either design time or run-time (e.g., regulations found in different countries and regions, products or services being delivered, or customer

* This work has been developed with the support of MICINN under the project EVERYWARE TIN2010-18011.

categories) [18, 4]. A collection of related process variants is denoted as *process family*. Examples can be found in almost every domain; e.g., [8] describes a process family for vehicle repair and maintenance with more than 900 process variants with country-, garage-, and vehicle-specific differences.

Properly dealing with process families constitutes a main challenge to reduce development and maintenance efforts in large process repositories. Designing and implementing each process variant from scratch and maintaining it separately would be inefficient and costly for companies. Thus, there is a great interest in capturing common process knowledge only once and re-using it in terms of *reference process models*, e.g., ITIL in IT service management, reference processes in SAP's ERP system, or medical guidelines. Even though respective proposals foster the reuse of common process knowledge, typically, they lack comprehensive support for explicitly describing variations [20]. In addition, they only provide limited support for run-time (re-)configuration and evolution (i.e., run-time flexibility), which inhibits the ability of an organization to respond to changes in an agile way. To deal with exceptions, uncertainty, and evolving processes, however, process families need to provide run-time flexibility as well [29].

In recent years, several proposals have been made to deal with process families. In the BP management field, model-driven techniques provide diverse solutions for managing process variants [23, 21, 8], i.e., for modeling, configuring, executing, and monitoring a process family. However, run-time flexibility and the evolution of process families have not been sufficiently considered so far. In turn, [9, 11] focus on flexibility issues at the execution level. Based on code injection, a process variant can be partially adapted to new environmental needs. However, respective techniques are difficult to apply for non-technical stakeholders and only cover parts of their flexibility needs. In the context of adaptive PAISs, in addition, solutions for enabling process flexibility are proposed [30, 31]. Despite the fact that these proposals are well suited for single process models, they cannot face the challenges raised by process model families. Finally, in the field of Software Product Lines, flexibility issues in product families are discussed [10], i.e., feature models allow specifying variations between members of a product family. However, these techniques focus on design time configuration, neglecting run-time configuration support.

In this paper, we characterize run-time flexibility needs of process families using two case studies for illustration purposes. We discuss open issues and research challenges regarding run-time (re-)configuration and evolution of process variants. Further, we provide a review of methods, technologies, and tools for BP variability, and discuss how they address respective run-time flexibility issues.

Section 2 presents two examples of process families used for illustration purpose. In Section 3, we define the main concepts for process families and introduce existing variability proposals. Section 4 analyzes run-time flexibility needs of process families. Finally, Section 5 summarizes the paper.

2 Examples of Process Families

To illustrate run-time flexibility needs of process families, we refer to process families from the healthcare and automotive domains, which we analyzed in two case studies. More precisely, our first family comprises more than 90 process variants for handling medical examinations, either standard (i.e., planned) or emergency (i.e., unplanned), in large hospitals [14]. In turn, our second process family consists of more than 20 variants dealing with product change management in the automotive domain.

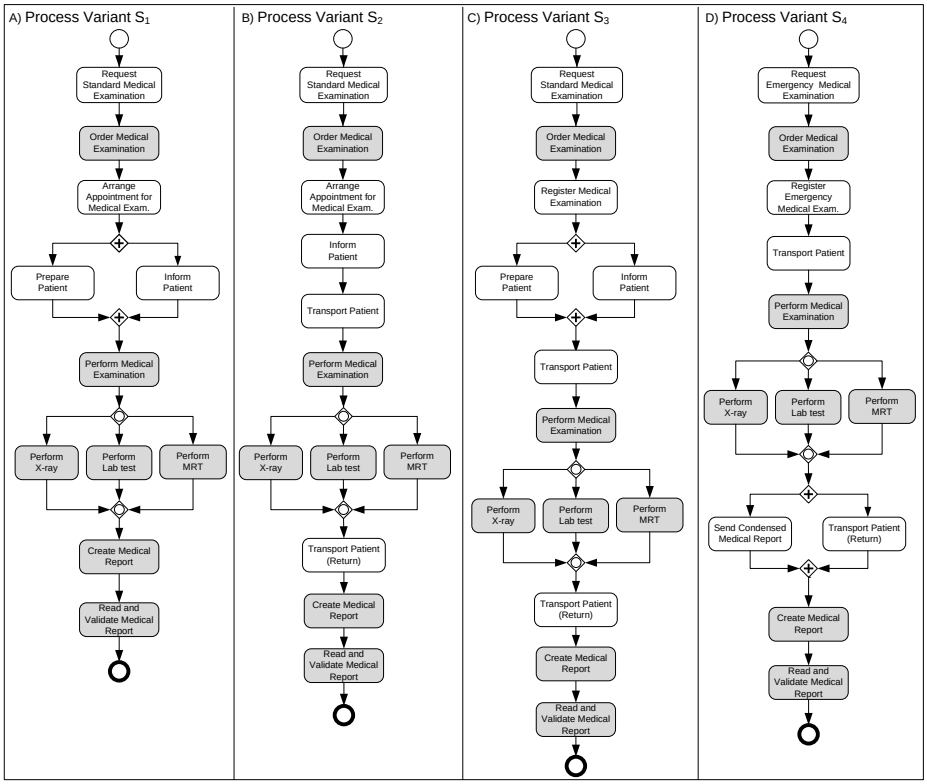

Fig. 1. Process Variants for Handling Medical Examinations

Process Family 1 (Handling Medical Examinations). Fig. 1 exemplifies four simplified process variants of a process family for handling medical examinations. These variants have several activities in common (highlighted in grey), e.g., *Order Medical Examination, Perform Medical Examination, Perform X-ray, Perform Lab Test, Perform MRT*, and *Create Medical Report*. However, the variants also show differences, e.g., in respect to the kind of examination (i.e., standard vs. emergency medical examination), the way the examination is scheduled (e.g., making and appointment or simply registering the examination),

or the need for the presence of specific activities depending on the given context and configuration settings (e.g., *Prepare Patient* or *Transport Patient*).

Process Family 2 (Product Change Management). In the automotive domain, product change management constitutes a complex process [6, 7] for which different variants exist, depending on the implementation costs and change impact, as well as the product phase during which the change is requested (e.g., development, start-up, or production).

3 Coping with Business Process Variability

This section provides basic notions related to BP variability (cf. Section 3.1) and introduces existing proposals for enabling it (cf. Section 3.2).

3.1 Basic Notions

When dealing with variability, it is important to define (1) what parts of the BP model may vary according to a specific context, (2) what alternatives fit in each of those parts, and (3) which conditions make these alternatives being selected. The first issue refers to the identification of the parts being subject to variation, which are commonly known as *variation points*. The second issue refers to the different alternatives that exist for these variation points, which we denote as *process fragment substitutions*. The third issue refers to the *context* in which these variations occur. Such context is usually represented by a set of variables gathered in a *context model* in which the BP model is used. When combining these variability aspects, we obtain a *configurable process model*, which is capable of representing the complete *process family*, i.e., collection of *process variants*.

Two major approaches are discussed in literature to define a *configurable process model*: behavioral and structural [18]. While a *behavioral approach* is based on a unique artifact integrating the behavior of all family members (i.e., process variants), a *structural approach* results in a set of artifacts, separately representing different aspects of the process family (e.g., commonalities captured in a *base process model* and variations captured in *change artifacts*). Despite these differences, configurable process models—irrespective of the approach used—allow *eliminating redundancies* by representing variant commonalities only once. Further, they allow fostering *model reuse*, i.e., model parts can be shared among multiple variants [26]. After creating the configurable process model, it must be *verified*, i.e., it has to be ensured that all derivable variants are syntactically correct. Additionally, the configurable process model must be *validated*, i.e., it must be ensured that the business requirements are properly reflected by the model. Given a configurable process model and taking the current context conditions into account, an *individualization process* is performed to derive a particular *process variant* [13]. According to the process enactment system chosen, the derived

process variant is then transformed such that it can be deployed on this system [5]. Fig. 2 shows how to move from the definition of a process family to the creation and execution of a *process variant instance*. It shows a traditional view when dealing with process families. However, this is not sufficient (as illustrated in Section 4) since run-time (re-)configuration and evolution of process families should be covered along the entire BP lifecycle as well.

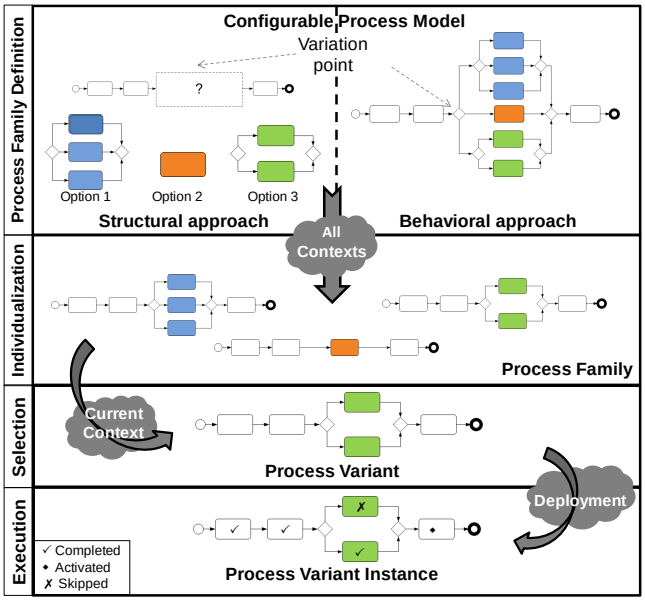

Fig. 2. From Process Family Definition to Process Variant Enactment

3.2 Existing Proposals Dealing with BP Variability

In literature, there are different proposals dealing with BP variability: PESOA [23], C-EPC [21], RULE (Rule representation and processing) [12], Provop [8], PPM (Partial Process Models) [16], and Worklets [1]. **PESOA** and **C-EPC** are both behavioral approaches for capturing variability in process families [23, 21]. To identify variation points in the configurable process model, PESOA defines a set of annotations related to the variable activities, while C-EPC makes use of *configurable functions* (i.e., activities) and *connectors* (e.g., OR gateway). The conditions that instantiate the alternatives for such variation points are defined through features or *configuration requirements*. In turn, **Provop** is a structural approach, i.e., a process variant is configured by applying a set of pre-defined *change operations* (i.e., change artifacts) to a base process model [8]. **RULE** is a behavioral approach that applies *business rules* (i.e., change artifacts) to configure process variants from a *process template* (i.e., base process model) [12]. In turn, **PPM** is a query-based approach where process variants combine their own concrete activities with behavior-inherited from parent processes [16].

Finally, **Worklets** allow handling exceptions at run-time through *dynamic reconfiguration*. A *worklet* is a complete workflow specification which, based on context conditions, handles one specific task in a composite parent process [1].

4 Run-time Flexibility in Process Model Families

In our context, flexibility represents the ability of a process family to change selected model parts, while keeping other model parts stable [24]. Due to the high dynamics in real-world environments, not all configurations can be made at design time. Thus, run-time flexibility arises as one of the core challenges for managing process families. Specifically, it requires the ability to deal with *pre-planned changes* and *dynamic evolution* (cf. Fig. 3) [30]. While the first issue refers to the *run-time configuration of process variants* (i.e., deferring the resolution of variation points to run-time) as well as their run-time *re-configuration* (i.e., switching between process variant models), the second issue deals with the *evolution of single process variants* (e.g., to copy with non-planned situations in a specific process variants) or the *evolution of the entire process family* (i.e., to deal with the re-design of the configurable process model). For each of these issues, we provide a general description, an illustrative example, a discussion of how existing proposals support them, and challenges to be tackled.

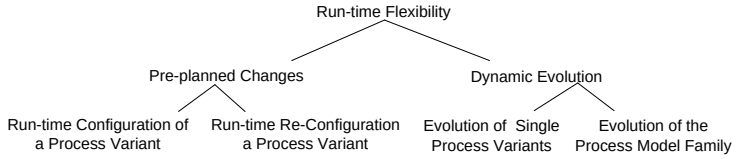

Fig. 3. Run-time Flexibility for Process Model Families

4.1 Run-time Configuration of Process Variants

General Description. As illustrated in Fig. 2, *process variant instances* are executed according to the schema of the process variant model configured at design time. However, certain configuration decisions (i.e., resolutions of variation points) can only be made during run-time when needed context information becomes available. Thus, techniques are required that allow deferring the resolution of variation points to the run-time. In addition, the subject of run-time configuration may refer to any modeling element (e.g., activities, resources, data, events, or operations); i.e., proper support for dynamically configuring arbitrary model elements during the execution of process variant instances is needed.

Example 3 (Run-time configuration of a process variant). In the medical examination process (cf. Process Family 1), the tests to be performed (i.e., *X-ray, MRT,* and *Lab tests*) only become known once the patient has been examined. Hence, their selection can only be done once process variant instances are

enacted. In addition, the role in charge of a treatment may change depending on the disease diagnosed. Finally, whether activities related to patient transportation are needed is decided during run-time depending on the status of the patient. As soon as this information becomes available, the process variant model of the respective instance should be configured accordingly, e.g., by removing the activities dealing with transportation.

Existing Support. Current variability proposals do not provide proper support for run-time configuration of process variants. Despite the fact that most proposals provide basic run-time support for process variants (e.g., execution of process variant instances), none of them allows for the inclusion of variation points in process variant models and their run-time configuration by end-users.

Challenges. One challenge is to ensure *soundness* of the (partially) dynamically configured process variant. Even though configuration is partially performed at run-time, soundness should be ensured at design time, i.e., for each process variant models its soundness should be guaranteed at design time even if parts of the model are dynamically configured at run-time. Existing proposals [28, 7] mainly focus on control flow, but have neglected other perspectives of process variants so far (e.g., resources, data flow, events). Another challenge is to decide by *whom, when,* and *based on which information* run-time configurations may be made. Sometimes, this might be accomplished automatically based on context information, which can be derived from process data, while in other cases user interactions are required. For the latter, intelligent user support at a high level of abstraction is needed. Finally, techniques for *visualizing dynamic configuration options* are needed.

4.2 Run-time Re-Configuration of Process Variants

General Description. Application context may dynamically change during run-time [25], making a re-configuration of a running process variant instance necessary to allow it to switch from the current process variant model to another one [27]. Unlike ad-hoc changes (i.e., unplanned changes) known from adaptive PAISs, re-configuration options are usually known at design time and hence can be captured in the configurable process model.

Example 4 (Run-time re-configuration of a process variant). In the context of product change management (cf. Process Family 2), for a particular car model, a change may be requested by a supplier in the *start-up* phase. Assume that during the processing of this change request, which may take several weeks or even longer, the car model switches to phase *production*. Then, the change request must be handled by a process variant model different from the one applied in the start-up phase; e.g., different procedures for estimating the

costs of the requested change and for approving it are needed. Therefore, the process variant instance needs to be executed according to a different process variant model, i.e., one must switch to another pre-specified process variant.

Existing Support. Run-time re-configuration of process variants is partially covered by existing proposals. In Provop, it is supported by including *variant branchings* in the configured process variant model and encapsulating the change operations within the variant branches. Worklets allow for run-time re-configuration since their instantiation is performed dynamically based on context changes. In this line, existing proposals for handling exceptions (e.g., exception handling patterns [22]) can be used for enabling run-time re-configurations of process variants. However, neither PESOA, nor C-EPC, nor RULE, nor PPM provide such re-configuration support. Finally, in the area of product families, software system run-time re-configuration is provided by using feature models [3]. By enabling/disabling features, systems dynamically switch from one feature configuration (i.e., a system configuration in terms of functionality) to another.

Challenges. Accurate information about the current context and upcoming context changes must be provided. Thus, *monitoring* as well as *prediction* techniques are needed. For this purpose, existing context monitors (e.g., ASTRO [2]) can be used to gather the required context information. In addition, switching from one process variant to another requires *sophisticated exception handling* beyond already existing techniques (e.g., to abort branches no longer needed [18]). Overall, run-time re-configuration should be supported in a controlled, efficient, and comprehensible manner [7, 6]. For example, *consistent process instance states* (including data consistency) must be ensured before continuing executing the process variant instance on the new process variant model.

4.3 Evolution of Single Process Variants

General Description. For many application scenarios, it is unrealistic to assume that all possible situations can be anticipated at design time and thus be incorporated into the configurable process model *a priori*. As a consequence, at run-time situations might emerge in which a process variant model no longer reflects the business case happening in the real world. In such a situation, authorized process participants should be allowed to evolve process variants models to realign their specification to the real-world business case. In addition, such evolution may also require the propagation of the changes to running process variant instances [30].

Example 5 (Evolution of a single process variant). For medical examinations, due to new regulations, every time the patient is transported, an extra

physical examination shall be performed. Thus, process variants including patient transportation need to be modified accordingly. For this, the variant model is evolved and changes are propagated to respective instances, if desired.

Existing Support. Existing variability proposals do not provide support for evolving process variants. However, in the area of adaptive PAISs, there are techniques enabling users to evolve process definitions and allowing for propagating changes to process variant instances [18]. Finally, there exist other proposals that allow adapting process variant instances at the execution level. For example, by injecting pieces of code, running process variant instances can be modified according to context changes [9, 11]. However, they are not suitable for evolving process variant models since they do not cover changes of the variant model.

Challenges. When evolving a single process variant, proper *change propagation* to running process variant instances is necessary. Note that this might be a complex task in case a process variant contains variation points that may be dynamically configured. In addition, changes in a single process variant model may require checking whether other process variants are affected as well. In the latter, the affected process variants need to be changed accordingly.

4.4 Evolution of the Process Family

General Description. To deal with environmental changes (e.g., changes of legal regulations), a process family must evolve at the schema level, i.e., changes of the configurable process model to address the changing requirements (e.g., by adopting additional variation points), increase its quality, or optimize its use. As a result, a new process family is obtained. In this context, co-existing schema versions of a configurable process model may have to be maintained.

Example 6 (Evolution of the process family). Due to newly emerging legal requirements for medical examinations, every patient needs to sign an extra document before being examined. Hence, the configurable process model must be modified since several process variants are affected by that change, leading to the evolution of the whole process family. In addition, the extra document is also relevant for patients for which the medical examination has already been started. Thus, evolving the process family requires the propagation of the changes to all configured process variants and—if desired—to running process variant instances.

Existing Support. C-EPC, Provop, and Worklets support the evolution of configurable process models, but not the propagation of respective changes to already configured process variants. In turn, in RULE, configurable process models can be evolved by adding new rules: changes are automatically propagated to process variants. Neither PESOA nor PPM support such an evolution. Support for handling different versions is provided by none of the proposals.

Challenges. Schema evolution of process model families may require the *propagation* of the changes to affected configured process variants and—if desired—to their running process variant instances [17]. In addition, this propagation should be performed correctly and efficiently. Furthermore, evolution may include new variation points for which the resolution time is deferred to run-time. Thus, their proper *run-time configuration* is required. Finally, *conflicts* between single process variants which have been individually evolved (cf. in Sect. 4.3), and the evolution of the configurable process model need to be handled. Even though similarities with evolution techniques known from adaptive PAISs exist [18], they cannot be directly applied for evolving configurable process models with their specific modeling elements and their dynamically configured parts. Therefore, different strategies for change propagation are needed.

4.5 Summary and Discussion

As discussed in Sections 4.1-4.4, when dealing with run-time flexibility in process model families, four issues are fundamental: *run-time configuration of process variants* and *re-configuration of process variants*, *evolution of single process variants*, and *evolution of the entire process model family*. Table 1 summarizes how existing variability proposals support them. For each issue, we differentiate between *no* [-], *partial* [+/-], and *full support* [+].

	PESOA	C-EPC	Provop	RULE	PPM	Worklets
Run-time conf. of process variants	-	-	-	-	-	-
Run-time re-conf. of process variants	-	-	+	-	-	+
Evolution of single process variants	-	-	-	-	-	-
Evolution of the process family	-	+/-	+/-	+/-	-	+/-

Table 1. Support for Run-time Flexibility Needs

An issue partially covered by existing BP variability proposals is run-time re-configuration of process variants. Applying different techniques, Provop and Worklets allow dynamic switches between process variants. However, our analysis has revealed that run-time configuration and evolution of single process variants are not well supported by any of the variability proposals. Regarding the evolution of configurable process models, existing proposals provide basic support, but lack advanced techniques for the controlled propagation of changes of the configurable process model to process variants and related instances. Our analysis has additionally shown that techniques from other areas provide partial solutions for addressing the flexibility needs discussed. For example, in the area of product families, support for the run-time re-configuration of software systems in terms of features is provided [3]. Based on feature models, systems may dynamically switch from one feature configuration to another one. However, this technique cannot be easily transferred to process model families since features cannot be directly mapped to BP specifications. Finally, adaptive PAISs provide techniques enabling users to evolve process schemes [18]. However, support going

beyond existing adaptive PAISs is needed for properly handling flexibility issues in process families.

Even though existing proposals have addressed specific aspects partially, holistic support for run-time flexibility (encompassing integrated support for run-time configuration, re-configuration of process variants, evolution of single process variants, and evolution of configurable process models) is still missing. In the context of run-time configuration, correctness of process variants, visualization, and authorization (i.e., who, when, and based on what information changes can be done) constitute, further challenges to be addressed. In addition, sophisticated exception handling techniques (e.g., abort branches or undo already performed activities) are needed to cope with run-time re-configuration of process variants. Finally, regarding the evolution of process families, techniques for propagating changes to already configured process variants are required.

5 Conclusions

In this paper, we describe run-time flexibility needs of process model families and provide an overview regarding the existing support of BP variability. Our research has revealed that holistic support for run-time flexibility in process families is still missing. Although support for modeling, configuring, and executing process variants is provided, dealing with pre-planned changes and evolution at run-time has not been well covered yet. Therefore, in a next step, we plan to introduce run-time flexibility in process families and develop a framework to support the dynamic evolution of configurable process models and process variants.

References

1. Adams, M.J.: Facilitating dynamic flexibility and exception handling for workflows. PhD thesis, Queensland University of Technology, (2007).
2. Barbon, F., Traverso, P., Pistore, M., Trainotti, M.: Run-Time Monitoring of Instances and Classes of Web Service Compositions. In Proc. ICWS, 63–71 (2006).
3. Cetina, C., Giner, P., Fons, J, Pelechano, V: Autonomic computing through reuse of variability models at runtime: the case of smart homes. Comp. 42(10), 37–43 (2009).
4. Dijkman, R., La Rosa, M., Reijers H.A: Managing large collections of business process models - Current techniques and challenges, Comp. in Ind. 63(2), 91–97 (2012).
5. Günther, C.W., Rinderle-Ma, S., Reichert, M., van der Aalst, W.M.P., Recker, J.: Using process mining to learn form process changes in evolutionary systems. Int. Journal of Business Process Integration and Management 3(1), 61–78 (2008).
6. Hallerbach, A., Bauer, T., Reichert, M.: Context-based configuration of process variants. In Proc. TCoB'08, 31–40 (2008).
7. Hallerbach, A., Bauer, T., Reichert, M.: Guaranteeing soundness of configurable process variants in Provop. In Proc. CEC'09, 98–105(2009).
8. Hallerbach, A., Bauer, T., Reichert, M.: Capturing variability in business process models: the Provop approach. Soft. Proc.: Impro. and Prac. 22(6–7), 519–546 (2010).

9. Hermosillo, G., Seinturier, L., Duchien, L.: Creating Context-Adaptive Business Process. In Proc. ICSOC'08, 228–242 (2010).
10. Kang, K.C., Cohen, S.G., Hess, J.A., Novak, W.E., Peterson, A.S.: Feature-oriented domain analysis (FODA) feasibility study. TR, Carniege-Mellon Univ.(1990).
11. Koning, M., ai Sun, C., Sinnema, M., Avgeriou, P.: VxBPEL: Supporting variability for web services in BPEL. Inf. and Soft. Tech. 51(2), 258–269 (2009).
12. Kumar, A., Wen, Y.: Design and management of flexible process variants using templates and rules. Int. Journal Computers in Industry 63(2), 112–130 (2012).
13. La Rosa, M., Dumas, M., ter Hofstede, A.H.M.: Modelling Business Process Variability for Design-Time Configuration. In Handbook of Research on Business Process Modeling, IGI Publisher, 204–228 (2009).
14. Li, C., Reichert, M., Wombacher, A.: Mining business process variants: challenges, scenarios, algorithms. Data & Knowledge Engineering 70 (5), 409–434 (2011).
15. Müller, D., Herbst, J., Hammori, M., Reichert, M.: IT support for release management processes in the automotive industry. In Proc. BPM'06, 368–377 (2006).
16. Pascalau, E., Awad, A., Sakr, S., Weske, M.: On Maintaining Consistency of Process Model Variants. BPM Workshops, 289–300 (2010).
17. Reichert, M., Rinderle, S., Dadam, P.: On the common support of workflow type and instance changes under correctness constraints. Proc. CoopIS, 407–425 (2003).
18. Reichert. M, Weber, B.: Enabling flexibility in process-aware information systems: challenges, methods, technologies. Springer (2012).
19. Reinhartz-Berger, I., Soffer, P., Sturm, A.: Organisational reference models: supporting an adequate design of local business processes. IBPIM 4(2), 134–149 (2009).
20. Reinhartz-Berger, I., Soffer, P., Sturm, A.: Extending the adaptability of reference models. IEEE Trans. on Sys., Man, and Cyb. 40(5), 1045–1056 (2010).
21. Rosemann, M., van der Aalst, W.M.P.: A configurable reference modeling language. Information Systems 32(1), 1–23 (2007).
22. Russell N., van der Aalst W.M.P., ter Hofstede A.: Exception handling patterns in process-aware information systems. In Proc. CAiSE'06, 288-302 (2006).
23. Schnieders, A., Puhlmann, F.: Variability mechanisms in e-business process families. In Proc. BIS'06, 583-601 (2006).
24. Soffer, P.: On the notion of flexibility in business processes. In Proc. CAiSE'05 Workshops, 35–42 (2005).
25. Soffer, P.: Scope analysis: identifying the impact of changes in business process models. Sof. Process: Impro. and Practice 10(4), 393–402 (2005).
26. Torres, V., Zugal, S., Weber, B., Reichert, M., Ayora, C., Pelechano, V.: A qualitative comparison of appraches supporting business process variability. BPM Workshops 2012 (to appear).
27. van der Aalst, W.M.P., Basten, T.: Inheritance of workflows: An approach to tackling problems related to change. Theo. Comp. Science 270(1–2), 125–203 (2002).
28. van der Aalst, W.M.P., Dumas, M., Gottschalk, F., ter Hofstede, A.H.M., La Rosa, M., Mendling, J.: Preserving correctness during business process model configuration. Formal Asp. Comput. 22(3–4), 459-482 (2010).
29. Weber, B., Reichert, M., Rinderle-Ma, S.: Change patterns and change support features - Enhancing flexibility in process-aware information systems. Data & Knoweldge Engineering 66(3), 438–466 (2008).
30. Weber, B., Sadiq, S.W., Reichert, M.: Beyond rigidity - dynamic process lifecycle support. Computer Science - R&D 23(2), 45–65 (2009).
31. Weber, B., Reichert, M., Reijers, H.A., Mendling, J.: Refactoring large process model repositories. Computers in Industry 62(5), 467–486 (2011).