Towards the Usage of Object-Aware Process Variants in Multiple Autonomous Organisations

Philipp Hehnle[1,*], Manfred Reichert[1,*]

[1]Institute of Databases and Information Systems, Ulm University, Germany

Abstract

Managing similar software products and thereby reducing costs has been subject to research in the field of Software Product Line Engineering (SPLE). In our previous work, we applied the concepts of SPLE to activity-centric processes. Although research was conducted to manage variants of object-aware processes, there are still open challenges. In this paper, we address the challenge of managing object-aware process variants in autonomous organisations, which run their information systems separately.

Keywords

Business Process Management, Process Configuration, Process Variability, Software Reuse

1. Introduction

As a result of the right to self-administration, various German municipalities use slightly different digitised variants of the same business processes [1]. Existing approaches for managing process variants focus on the control flow rather than on the implementation level, i.e. in a process model, fragments may be added, moved, or removed [2] and gateways may be restricted (e.g. an OR gateway may be restricted to an AND gateway) [3]. To reduce costs when developing similar software products, the discipline of Software Product Line (SPL) Engineering emerged [4]. An SPL comprises a set of common core software artefacts from which individual software products may be derived, i.e. be configured by selecting and combining the software artefacts [5][6]. In our previous work [1], we applied the concepts of SPL Engineering to activity-centric process management to benefit from the advantages (e.g. reduced costs) in that a process-aware information systems may be derived from a set of software artefacts including a process model by selecting different implementations for the activities of that process model. Thereby, it is possible to derive various process-aware information systems with varying activity implementations.

Frequently, in activity-centric process management, data objects are under-specified. Besides, activity-centric process management lacks flexibility as users must not carry out activities in varying sequences. In contrast, data-centric process management [7] is a paradigm specifying unstructured or semi-structured processes in which data and processes are coupled tightly and the progress of a process is driven by data. The object-aware process management approach [8]

ZEUS'24: 16th Central European Workshop on Services and their Composition, February 29 - March 1, 2024, Ulm, Germany

*Corresponding author.

philipp.hehnle@uni-ulm.de (P. Hehnle); manfred.reichert@uni-ulm.de (M. Reichert)

0009-0006-0420-7000 (P. Hehnle); 0000-0003-2536-4153 (M. Reichert)

S. Böhm and D. Lübke (Eds.): 16th ZEUS Workshop, ZEUS 2024, Ulm, Germany, 29 February–1 March 2024, published at http://ceur-ws.org

implements the data-centric process management paradigm. There are object types and their corresponding object behaviours. The latter are specified in terms of lifecycle processes which comprise states and state transitions. At runtime, the object types are instantiated with their lifecycle processes. Activities with user forms are generated when a state is reached in which data is required and activities are completed (i.e. the process continues) when the required data becomes available (i.e. the user has entered the object attributes). Thus, the progress of a lifecycle process is data-driven and determined by changes of the object attributes. PHILharmonicFlows is a framework for object-aware process management. In [9], [10], and [11], an object-aware and process-aware information system (OPAIS) is presented as a proof-of-concept implementation of PHILharmonicFlows. While there have been approaches dealing with variants in activity-centric processes, only little research covers variants in OPAISs. The work in [12] deals with variants in one OPAISs. However, the same process can be operated in variants in different autonomous organisations that run their OPAIS separately. The management of process variants in separate OPAISs remains an open challenge, which is addressed in this paper.

2. Related Work

This section presents an approach to deal with variants in OPAISs as well as core concepts of SPL Engineering.

In [12], an approach is presented for coping with process variants in OPAISs. At design time, when evolving a process in an OPAIS (e.g. adding or removing attributes from an object or updating a lifecycle process) the changes are logged rather than propagated to the process in production. As soon as the changes need to be deployed to production, the logs created during design time can be replayed, i.e. propagating the changes to production. When developing various variants of a process, the base process containing the commonalities is copied by replaying the corresponding logs. Each copy of the base process is the starting point for a variant where the variant-specific changes are applied to. When the base process is altered, the changes can be applied to all variants by replaying the logs triggered by the changes to the base process. Certainly, this approach facilitates the development of variants and the propagation of common changes. However, the approach neglects that autonomous organisations that run process variants will most likely have physically independent hardware/OPAIS instances. Therefore, the logs need to be distributed among various OPAIS instances. Furthermore, the approach assumes that each variant is used once. However, multiple organisation might use the same variant each on their separate OPAIS instance. The logs should not be copied but referenced so that changes to a variant can be performed centrally and applied to all organisations that use the same variant. Moreover, an organisation might use a combination of variants instead of creating a new variant from scratch or use an outdated version of a variant while the variant has been updated centrally, i.e. different versions of a variant might be in production. During bug analysis, it becomes necessary to rebuild a specific version of a variant of an organisation.

Feature models [13] describe software products in terms of their features by outlining which features are mandatory, optional, and which features require or exclude each other.

Configuration management in SPL Engineering [14] deals with managing the versions of the common core artefacts as well as the derived software products over time. In contrast, version

control tools for configuration management in software engineering only manage one software product over time. In [15], a Divide & Conquer approach for configuration management is proposed that divides the challenge in sub-challenges and solves them with the use of existing version control tools from software engineering.

3. Research Direction

This section presents an approach to deal with variants in OPAIS of autonomous organisations. First, a real-world example is described which serves as motivation. Then, requirements are elicited based on the example. Finally, an approach satisfying these requirements is sketched.

During a cooperation with German municipalities, the process for checking the special parking permit application of craftspersons was studied. This process is pictured in a slightly adapted and simplified form compared to our previous work [1]. In German municipalities craftspersons may apply for a special parking permit which allows them to park in zones of the city where citizens have to pay or where parking is not permitted. The object-aware process for checking this application is depicted in Figure 1. The applicant must provide information about her company and the company's cars for which the parking permit shall be valid (cf. object types on the left-hand side). On the right-hand side, the lifecycle processes for the object types are shown. Different municipalities might need adaptions to the base process (cf. Figure 1) which results in variants. Some municipalities might require the applicant to submit pictures of the cars to prevent issuing parking permits for private cars. In other municipalities, information about the cars is not required at all. Then, the parking permit is valid for whatever car it is situated in. German municipalities are autonomous organisations, which run their information systems separately.

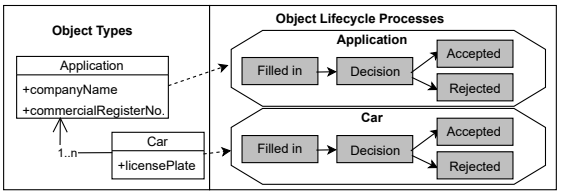

Figure 1: Object-Aware Process

Based on the example, the requirements are deduced for managing variants in OPAISs of autonomous organisations.

R1: Each organisation must be able to select a combination of adaptions in conjunction with the base process and build (i.e derive) a process variant.

R2: Multiple organisations may derive the same process variant.

R3: Adaptions need to be managed centrally in that changes to them can be propagated easily to all process variants that are using these adaptions.

R4: Not every combination of adaptions may be valid. It needs to be evident what adaptions exclude each other.

R5: As the base process and the adaptions evolve, not every organisation may want to apply the updates.

R6: For bug analysis, it needs to be evident which organisation uses which version and adaptions of the base process in order to rebuild and analyse the corresponding variant.

Using concepts of SPL Engineering, namely feature models [13] and the Divide & Conquer approach for configuration management proposed in [15], it becomes possible to satisfy the

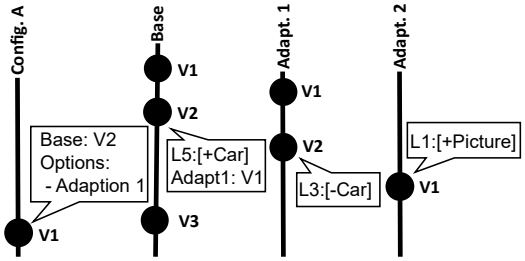

Figure 2: Configuration Management

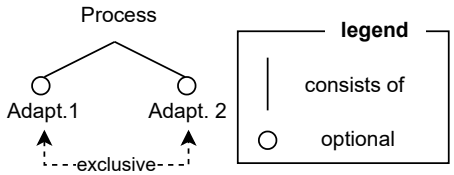

Figure 3: Feature Diagram

requirements. The base process and every adaption are stored as logs each in a central repository under version control. Thereby, changes due to the evolution of the base process and the adaptions can be applied centrally. Every organisation has a configuration that stores the version of the base process and the selected adaptions. During derivation, the referenced logs are fetched and replayed in the organisation's OPAIS instance. As each repository is under version control, older version of the logs can be used which allows the organisations to stay on older versions of the variants. Furthermore, for bug analysis, the configuration of each organisation is under version control as well. Consequently, to analyse a bug a specific version of a variant of an organisation can be restored and inspected. Figure 2 is an example for configuration management and shows the repositories and their version history over time. It reveals that Adaption 1 in Version V2 uses log L3 to remove the object type *Car*, whereas Adaption 2 in Version V1 uses log L1 to add the object type *Picture*. Obviously, Adaptions 1 and 2 cannot be co-selected as the picture object type needs a car object type it belongs to, which is removed in Adaption 1. Figure 3 imposes the base process with its adaptions in a feature model, which indicates that Adaptions 1 and 2 are mutually exclusive. Furthermore, Figure 2 represents the evolution of the base process model (viz. three versions) and the exemplary repository of one organisation, which stores a configuration (Config. A) to derive a process variant by specifying that the base process in version V2 and Adaptation 1 is selected.

4. Conclusion and Outlook

This paper deals with managing variants in OPAISs in that multiple autonomous organisations may derive process variants from a base process by selecting and combining adaption to the base process whereas different versions of both the base process and the adaptions may be used. Consequently, there may be a variety of process variants and versions that need to be tracked. This work presents an approach that applies concepts of SPL Engineering to OPAIS variants in order to keep track of the process variants and their versions.

In future work, we plan to assemble a tool chain that is able to automatically derive and keep track of process variants in OPAIS by using, extending, and integrating tools from SPL Engineering, and, where necessary, creating new tools. Furthermore, black-box-activities [9] implementing business logic as well as customised forms (i.e. opposed to generated forms) need to be considered in the future.

References

[1] P. Hehnle, M. Reichert, Handling Process Variants in Information Systems with Software Product Line Engineering, in: 2023 IEEE 25th Conference on Business Informatics (CBI), 2023, pp. 1–10.

[2] A. Hallerbach, T. Bauer, M. Reichert, Capturing variability in business process models: the Provop approach, Journal of Software Maintenance and Evolution: Research and Practice 22 (2010) 519–546.

[3] W. M. P. van der Aalst, A. Dreiling, F. Gottschalk, M. Rosemann, M. H. Jansen-Vullers, Configurable Process Models as a Basis for Reference Modeling, in: Business Process Management Workshops, volume 3812 of *Lecture Notes in Computer Science*, Springer, 2006, pp. 512–518.

[4] L. Bass, P. Clements, R. Kazman, Software architecture in practice, Always learning, 3. ed. ed., Addison-Wesley, 2013.

[5] S. Deelstra, M. Sinnema, J. Bosch, Product derivation in software product families: A case study, Journal of Systems and Software 74 (2005) 173–194.

[6] C. Kästner, S. Apel, Feature-Oriented Software Development, in: Generative and Transformational Techniques in Software Engineering IV: International Summer School, GTTSE 2011, Springer, 2013, pp. 346–382.

[7] S. Steinau, A. Marrella, K. Andrews, F. Leotta, M. Mecella, M. Reichert, DALEC: A framework for the systematic evaluation of data-centric approaches to process management software, Software & Systems Modeling 18 (2019) 2679–2716.

[8] V. Künzle, M. Reichert, PHILharmonicFlows: towards a framework for object-aware process management, Journal of Software Maintenance and Evolution: Research and Practice 23 (2011) 205–244.

[9] C. M. Chiao, V. Kuenzle, K. Andrews, M. Reichert, A Tool for Supporting Object-Aware Processes, in: 2014 IEEE 18th International Enterprise Distributed Object Computing Conference Workshops and Demonstrations, 2014, pp. 410–413.

[10] K. Andrews, S. Steinau, M. Reichert, A Runtime Environment for Object-Aware Processes, in: International Conference on Business Process Management, 2015, pp. 6–10.

[11] S. Steinau, K. Andrews, M. Reichert, A Modeling Tool for PHILharmonicFlows Objects and Lifecycle Processes, in: International Conference on Business Process Management, 2017.

[12] K. Andrews, S. Steinau, M. Reichert, Enabling Process Variants and Versions in Distributed Object-Aware Process Management Systems, in: Information Systems in the Big Data Era, Springer, 2018, pp. 1–15.

[13] K. C. Kang, S. G. Cohen, J. A. Hess, W. E. Novak, S. Peterson, Feature-Oriented Domain Analysis (FODA) Feasability Study, Technical Report, 1990.

[14] P. Clements, L. Northrop, Software product lines: Practices and patterns, SEI series in software engineering, 7. print ed., Addison-Wesley, 2009.

[15] C. W. Krueger, Variation Management for Software Production Lines, in: Software Product Lines, volume 2379 of *Lecture Notes in Computer Science*, Springer Nature, 2002, pp. 37–48.