Alignment of Process Lifecycle and Software Product Line Engineering Phases

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

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

Abstract

Different organisation often run variants of the same business process. As opposed to managing each variant separately, variants can be maintained centrally in a customisable process model, which allows applying changes centrally and propagating them to each variant automatically. However, these approaches focus on the control flow of the business processes. Software Product Line Engineering (SPLE) deals with the efficient development of similar software products by selecting features for individual software variants. In previous work, we applied the concepts of SPLE to business processes in order to be able to select the implementation of certain process variants. For both business process management and SPLE there are a lifecycle and phases, respectively, that describe the development process, which is associated with concepts and tools to facilitate it. When combining process variability on the control flow level with process implementation variability using SPLE, the process lifecycle needs to be aligned with the phases of SPLE. In this work, we present a consolidated lifecycle that allows for the integrated usage of concepts and tools of process variability and SPLE.

Keywords

Business Process Management, Software Product Line Engineering, Process Configuration, Process Variability, Process Family, Software Reuse

1. Introduction

Different organisations run variants of the same business processes. Craftspersons may apply for a special parking permit in various German municipalities (e.g. Munich[1] and Stuttgart[2]), which permits them to park their cars in the urban area when visiting their clients without acquiring a parking permit for each stop. The process of applying and checking the application for a special parking permit is similar, yet slightly different among the municipalities. Process variants in the public sector have been identified by other researchers as well [1]. Different approaches propose *customisable process models* containing all process variants in one model forming a family of business process variants (process family) [1]. By using a customisable process model, changes and optimisations can be applied centrally, which reduces redundancies and thereby effort [2]. Special modelling languages and extensions to existing modelling languages have been proposed to model variability in business processes and tools have been presented to create a process variant by applying transformations to a customisable process model, which is called *deriving* a process variant [1]. For instance, the *Hide & Block* approach allows hiding and blocking edges of a customisable process model, which removes single edges or entire outgoing paths when deriving a process variant [3]. However, the approaches of customisable process models focus on control-flow variability of the business processes.

To avoid redundancies and thereby reduce costs when developing similar software products the discipline of Software Product Line Engineering has evolved. Software products can be built by selecting features from a set of common core artefacts, which constitute the Software Product Line (SPL) [4]. In literature, selecting features and building a software product is referred to as *deriving* a software product from an SPL [5, 6]. Preprocessors, aspect-oriented programming, special language extensions

Vienna'25: 17th Central European Workshop on Services and their Composition (ZEUS), February 20–21, 2025, Vienna, Austria

*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)

[1]https://stadt.muenchen.de/service/info/hauptabteilung-i-sicherheit-und-ordnung-praevention/1072021/
[2]https://www.stuttgart.de/organigramm/leistungen/sonderparkausweise-fuer-gewerbetreibende-und-soziale-dienste.php

like Jak for Java, and tool chains like FeatureHouse can be used to implement variability and derive software products by selecting and composing features from the common core artefacts of an SPL [6].

A software product that executes a business process in form of a process model is called Process-Aware Information System (PAIS) [7]. In previous work [8, 9], we applied the concepts and tools of SPL Engineering to PAISs to allow for implementation variability of business processes.

1.1. Problem statement

Both the approaches for customisable process models and SPL Engineering have defined steps, which build a lifecycle model and a phases model, respectively. Each step/phase consists of activities and is associated with certain concepts and tools supporting the necessary activities. In order to allow for process variability on the control-flow and implementation level the steps of the lifecycle and the phases of SPL Engineering need to be aligned, i.e. the associated concepts need to be linked and the tools of each step need to be integrated.

1.2. Contribution

This paper presents a consolidated lifecycle for managing and improving process variants in terms of control-flow and implementation variability whose steps are associated with the corresponding concepts and tools.

1.3. Outline

In Section 2, related work is assessed regarding the lifecycle of processes and the phases of SPL Engineering. Section 3 presents a consolidated lifecycle that integrates the steps of the process lifecycle and the phases of SPL Engineering including the concepts and tools of both domains. The paper concludes with a summary and an outlook in Section 4.

2. Related Work

This section presents the phases of SPL Engineering as well as the BPM lifecycle.

2.1. Phases of Software Product Line Engineering

SPL Engineering typically is divided into four phases [6, 10]. Figure 1 shows the phases of SPL Engineering. During *domain analysis*, the requirements in terms of features of the SPL are elicited including the commonalties and differences among the software products resulting in a feature model. Feature models were first introduced by [11] as a tree structure. Figure 5 shows an example feature model. The SPL is the root. The features the SPL consists of are connected via edges to the root. Features themselves can consists of features. Furthermore, features can be marked optional, mandatory, and alternative. During *domain implementation*, the features identified during *domain analysis* are implemented using variability mechanisms (e.g. preprocessor, aspect-oriented programming, language extensions like Jak, tool chains like FeatureHouse) that allow composing them dynamically. *Domain analysis* and *domain implementation* are considered *domain engineering* as they comprise activities concerning the SPL as a whole. When deriving a software product from an SPL, first, during *requirements analysis* the features that shall be contained in the software product are selected from the feature model created during domain analysis. The selected features form a configuration, which is used during *software generation.* Based on the configuration, the implemented features are composed and the software product is thereby generated. *Requirements analysis* and *software generation* are considered *application engineering* as they comprise activities concerning one individual software product that is derived from an SPL.

FeatureIDE is an integrated development environment (IDE) that supports developing SPLs during the corresponding phases [10]. For *domain analysis*, FeatureIDE provides a feature modelling tool (cf.

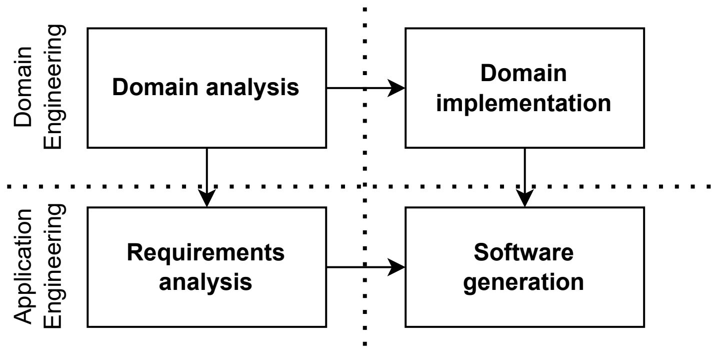


Figure 1: Phases of Software Product Line Engineering adapted from [6]

Figure 5). For *domain implementation*, FeatureIDE support various variability mechanisms. In order to select the desired features during *requirements analysis*, FeatureIDE provides a configuration editor (cf. Figure 6). During *software generation*, FeatureIDE relies on the chosen variability mechanism to compose the software product that shall be derived from the SPL.

2.2. Business Process Management Lifecycle

Business Process Management (BPM) supports "business processes using methods, techniques, and software to design, enact, control, and analyze operational processes involving humans, organizations, applications, documents and other sources of information."[12] Various researchers describe the steps and activities for BPM as BPM lifecycles.

While there are minor differences among the presented BPM lifecycles, the lifecycle models do have a lot in common. Figure 2 shows a consolidated BPM lifecycle that incorporates the commonalties of the presented BPM lifecycle models. During the *process design* step, the processes are modelled [13, 14, 15, 16]. A graphic representation of the business processes facilitates communication for stakeholders. This step is also called *modeling* [17, 18]. In the *process implementation* step, the process is realised in an organisation, usually it is implemented as software [16, 18]. A workflow-management-system can be used. This step is also called *configuration* [13, 14, 15]. In the context of process variability, [17] propose an additional step *selection/instantiation*, in which the desired process variant is selected and automatically instantiated while ensuring consistency and correctness of the instantiated process variant. Although this step is not included by other BPM lifecycle models, we incorporate it in our consolidated BPM lifecycle as we deal with process variants. In the *process enactment* step, the implemented process is executed, often in a workflow engine [13, 14]. This step is also called *execution* [15, 16, 18]. During *process evaluation*, running process instances are monitored and evaluated, data of running processes is aggregated to find deficiencies, and finally improvements are collected, which serve as input for the process *design step* in which the improvements are incorporated into the process model [16, 14]. This step is also called *diagnosis* [13, 15] or *optimisation* [17, 18].

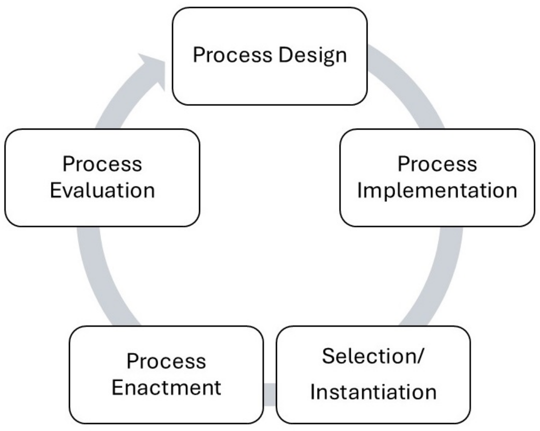


Figure 2: Consolidated BPM Lifecycle

Some researchers like [16, 18] propose further steps, e.g. strategic activities in which organisational and process goals are defined. These steps are out of scope for this work, as we focus on the digitisation of business processes.

The interested reader is referred to the literature reviews [19] and [20] about BPM lifecycles.

3. A Consolidated Lifecycle for Control-Flow and Implementation Variability of Processes

This section presents a consolidated lifecycle model that maps the phases of SPL Engineering to the steps of the BPM lifecycle in order to allow for process variability on the control-flow and implementation level. First, a real-world example is described based on which requirements are deduced. Finally, the consolidated lifecycle model is presented satisfying the requirements.

3.1. Special Parking Permit for Craftspersons

During a cooperation with German municipalities, the process for checking the special parking permit for craftspersons was inspected. See Example 1 as a simplified description of the process.

Example 1 In various German municipalities, Craftspersons can apply for a special parking permit, which allows them to park in the urban city during their customer visits without acquiring a parking permit for each stop. When a craftsperson has applied for a parking permit, the application needs to be checked. In some municipalities, a clerk checks the application, whereas in other municipalities the application check is carried out automatically. If the application is justified, the parking permit is issued. Otherwise, the craftsperson is notified of the rejection. The notification may be via mail, e-mail, or SMS.

In previous work [8, 9], we implemented Example 1 as a proof-of-concept whereas activities `check application` and `notify craftsperson` are variable, i.e their implementation may vary. We refer to a digitized business process realised as PAIS with variable activity implementations from which concrete variants may be derived as *PAIS Product Line*. The activity implementations were developed as self-contained plugins. When deriving a variant from the PAIS Product Line, we use the framework and tool chain *FeatureHouse* [21] to compose the selected plugins into one software artefact. Then, the plugins are registered with the corresponding activities they belong to.

Furthermore, it is conceivable that for some municipalities Example 1 is extended in that the craftsperson is notified either way (both when the permit is issued and when the application is rejected). Consequently, the control-flow of the process is variable as in some variants there is a notification activity when a parking permit is issued whereas in others there is not.

The described scenario represents a PAIS Product line whose control-flow is variable as well as its activity implementations.

3.2. Requirements

During the development of SPLs and customisable process models certain steps, which are associated with concepts and tools, need to be carried out, which form the phases of SPL Engineering and the BPM lifecycle, respectively. When implementing a PAIS Product Line comprising a process model with a variable control-flow and activities whose implementations may be exchanged, the steps related to the development of SPLs and customisable process models need to be aligned including the associated concepts and tools, creating a consolidated lifecycle. Consequently, the consolidated lifecycle needs to meet the following requirements:

R1: The phases of SPL Engineering and the steps of the BPM lifecycle need to be aligned, i.e. each phase need to be mapped to a step in order to align the related tasks that need do be carried out during the development of an SPL and a customisable process model.

R2: The concepts associated with the phases and steps of SPL Engineering and the BPM lifecycle need to be aligned, i.e. need to be connected.

R3: The tools supporting the phases of SPL Engineering and the steps of the BPM lifecycle need to be integrated, so that there is a holistic tool chain facilitating the development of a PAIS Product Line with a variable control-flow and variable activity implementations.

3.3. The Consolidated Lifecycle

In the following, the consolidated lifecycle, which aligns the phases of SPL Engineering and the steps of the BPM Lifecycle, is presented (cf. Figure 3).

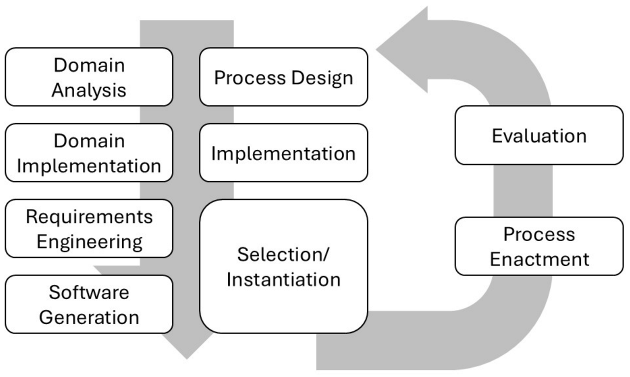


Figure 3: Consolidated Lifecycle for Control-Flow and Implementation Variability of Processes

Process models created in BPMN 2.0 serve as simple means of communication of the represented process for business analysts as well as developers who will implement the process [22]. Besides the visual representation, BPMN 2.0 process models have an XML representation, which allows for the execution of the models. Consequently, the process model, which will be implemented and executed, represents the requirements of the digitized business process. In SPL Engineering, the requirements are elicited during *domain analysis*, which results in a feature model. Consequently, the *domain analysis* can be mapped to the BPM lifecycle step *process design*. In previous work, we also mapped feature models to process models. Each activity, which may be variable (i.e. its implementation may be changed or the entire activity may be optional) equates to a feature in the feature model. Therefore, in previous work [8, 9], we introduced *process feature models*, which represent a PAIS Product Line as a feature model in three levels. The first level corresponds to the PAIS Product Line itself. The second level contains all activities whereas the third level comprises the possible implementations of the activities. Figure 4 shows an example process consisting of a start event, two activities, and an end event. The feature model in Figure 5 represents the example process and is created with FeatureIDE. It can be seen that for both Activities 1 and 2 there are two optional implementations.

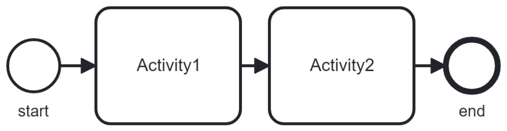


Figure 4: Example Process

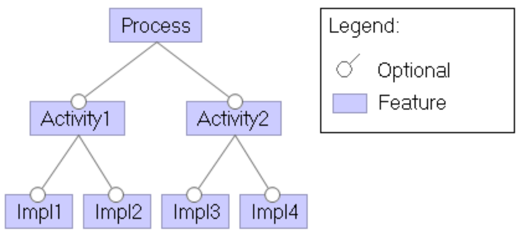
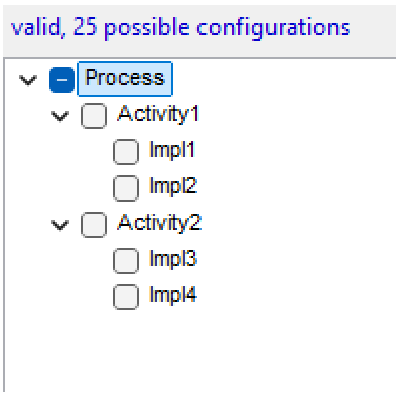


Figure 5: Feature Model in FeatureIDE

In the *domain implementation* phase, the requirements of the SPL are implemented. The BPM lifecycle has a similar step *implementation* in which the process model, which represents the requirements, is implemented in that it can be executed by a workflow-engine. Thus, *domain implementation* and *implementation* of the BPM lifecycle can be mapped. As introduced in our previous work [9], the activity implementations are developed as composable plugins.

During *requirements analysis*, the desired features are selected from the feature model whereas during the *selection* step of the BPM lifecycle the control-flow of the process is determined, for example by removing optional activities (Hide & Block approach [3]). The configuration editor of FeatureIDE can be used to select the desired implementations for the corresponding activities or if no implementation for an activity is chosen, the activity may be removed. Figure 6 shows the configuration editor in FeatureIDE that allows selecting activity implementations for example process depicted in Figure 4 taking into account the options laid out by the feature model in Figure 5.

The configuration of the selected features from the step *requirements analysis* and *selection* is used during *software generation* and *instantiation* to generate a PAIS including the process model with the selected activity implementation. In line with our previous work [9], the selected implementations may be composed to one artefact using FeatureHouse and register as plugins with the corresponding activities. Furthermore, optional activities for which no implementation was selected are removed from the process model, e.g. by applying the Hide & Block approach.


valid, 25 possible configurations

- Process
  - Activity1
    - Impl1
    - Impl2
  - Activity2
    - Impl3
    - Impl4

Figure 6: Configuration Editor in FeatureIDE

While the phases of SPL Engineering do not comprise a runtime perspective, the BPM lifecycle considers *process enactment* and *evaluation*. As our consolidated lifecycle model shall cover the development, maintenance, and improvement of PAIS Product Lines, we incorporate the runtime perspective of the BPM lifecycle. During *process enactment*, process instances are started and during *evaluation* the running process instances are monitored for flaws and bottlenecks, which will be analysed and used as input for improvement during the first step *domain analysis* and *process design*.

4. Conclusion

When implementing similar software products as SPL, certain tasks need to be carried out, which involve concepts and tools. These tasks are organised as the phases of SPL Engineering. The BPM lifecycle describes the tasks that need to be carried out in order to develop and maintain process variants on the control-flow level. When implementing process variants whose implementations as well as the control-flow vary, the approaches of SPL Engineering and customisable process models need to be combined. In this work, we presented a consolidated lifecycle model that aligns the tasks of SPL Engineering and the BPM lifecycle. Furthermore, the corresponding concepts were aligned and the tools where integrated for a holistic perspective and usage.

Future work shall reveal in a deep dive how the approaches for customisable process models (e.g. Hide & Block approach) can technically be integrated with the tools of SPL Engineering like the configuration editor of FeatureIDE and be used to automatically derive the process model during *software generation* and *instantiation* when using variability mechanisms like FeatureHouse.

Declaration on Generative AI

The authors have not employed any Generative AI tools.

References

[1] M. La Rosa, W. M. P. van der Aalst, M. Dumas, F. P. Milani, Business Process Variability Modeling: A survey, ACM Computing Surveys 50 (2017) 1–45.

[2] A. Delgado, D. Calegari, F. García, B. Weber, Model-driven management of BPMN-based business process families, Software & Systems Modeling 21 (2022) 2517–2553.

[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., 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, Springer, 2013, pp. 346–382.

[7] M. Dumas, W. van der Aalst, A. Ter Hofstede, Process-aware information systems: Bridging people and software through process technology, Wiley, 2005.

[8] 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.

[9] P. Hehnle, M. Reichert, Flexible process variant binding in information systems with software product line engineering, 2024. URL: https://arxiv.org/abs/2410.17689. `arXiv:2410.17689`, preprint.

[10] T. Thüm, C. Kästner, F. Benduhn, J. Meinicke, G. Saake, T. Leich, FeatureIDE: An extensible framework for feature-oriented software development, Science of Computer Programming 79 (2014) 70–85.

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

[12] W. M. P. van der Aalst, A. H. M. ter Hofstede, M. Weske, Business Process Management: A Survey, in: Business process management, volume 2678 of *Lecture Notes in Computer Science*, Springer, 2003, pp. 1–12.

[13] W. M. P. van der Aalst, Business process management: a personal view, Business Process Management Journal 10 (2004).

[14] M. Weske, Business Process Management, Springer, 2019.

[15] M. Netjes, H. Reijers, W. M. P. van der Aalst, Supporting the BPM life-cycle with FileNet, in: Proceedings of the 11th International Workshop on Exploring Modeling Methods for Systems Analysis and Design, 2006, pp. 135–146.

[16] M. zur Muehlen, D. T.-Y. Ho, Risk Management in the BPM Lifecycle, in: Business Process Management Workshops, volume 3812 of *Lecture Notes in Computer Science*, Springer, 2006, pp. 454–466.

[17] A. Hallerbach, T. Bauer, M. Reichert, Managing Process Variants in the Process Lifecycle, in: 10th Int'l Conf. on Enterprise Information Systems (ICEIS'08), 2008, pp. 154–161.

[18] C. Houy, P. Fettke, P. Loos, Empirical research in business process management – analysis of an emerging field of research, Business Process Management Journal 16 (2010) 619–661.

[19] R. Macedo de Morais, S. Kazan, S. Inês Dallavalle de Pádua, A. Lucirton Costa, An analysis of BPM lifecycles: from a literature review to a framework proposal, Business Process Management Journal 20 (2014) 412–432.

[20] N. Nousias, G. Tsakalidis, K. Vergidis, Not yet another BPM lifecycle: A synthesis of existing approaches using BPMN, Information and Software Technology 171 (2024) 107471.

[21] S. Apel, C. Kastner, C. Lengauer, FEATUREHOUSE: Language-independent, automated software composition, in: 2009 IEEE 31st International Conference on Software Engineering, 2009, pp. 221–231.

[22] Object Management Group, Business Process Model and Notation (BPMN): Version 2.0.2, 2013. URL: https://www.omg.org/spec/BPMN/2.0.2/PDF.