Software Product Line Adoption – Guidelines from a Case Study

Pasi Kuvaja[1], Jouni Similä[1], and Hanna Hanhela[2]

[1]Department of Information Processing Science, University of Oulu,
FI-90014 University of Oulu, Finland, jouni.simila@oulu.fi pasi.kuvaja@oulu.fi
[2]Digia, Sepänkatu 20, FI-90100 Oulu, hanna.hanhela@digia.com

Abstract. It is possible to proceed with software product line adoption only once without major reinvestments and loss of time and money. In the literature, reported experiences of using the adoption models are not to be found, and especially the suitability of the models has not been reported. The purpose of this research is to compare known adoption models by formulating general evaluation criteria for the selection of an adoption model. Next an adoption model is selected for empirical research based on the context of a multimedia unit of a global telecommunication company. The empirical part consists of a case study analyzing the present state of adoption and producing plans for proceeding with the adoption. The research results can be utilized when selecting an adoption model for an empirical case and adopting a software product line in a software intensive organization.

Keywords: software product line, adoption, adoption model, adoption strategy, guidelines

1. Introduction

Over the last decade, software product line engineering has been recognized as one of the most promising software development paradigms, which substantially increases the productivity of IT-related industries, enables them to handle the diversity of global markets, and reduces time to market [1]. In addition, the software product line approach can be considered as the first intra-organizational software reuse approach that has proven to be successful [2] and is a key strategic technology in attaining and maintaining unique competitive positions [3]. Software product line is “a set of software intensive systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way” [4]. Thus, different systems involving a product line are built by exploiting existing core assets. However, all of the existing core assets are not necessary to be used in one system.

Transition from conventional system development mode towards product line engineering requires adoption of a new approach. In software product line adoption, an organization changes its operational mode to develop product lines consisting of several products instead of developing products separately in-house. Adopting the

new approach is not, however, effortless. During the adoption, planning and coordinating of technical, management, organizational, and personnel changes are required [2, 5]. Furthermore, an adoption can be made either starting a product line from scratch or by exploiting existing systems [6, 7, 8, 9, and 10]. If the former strategy is used, needed changes are even larger than when using the latter strategy.

There are stories about successful adoption. After adopting a software product line, an organization can benefit in many different ways. Many studies have reported, that development time has shortened as efficiency has increased, less personnel to produce more systems is needed, more software is reused, overall and maintenance costs have decreased, and defects have reduced without compromising customer satisfaction [11,12,13,14,15]. Among the successful stories of software product line adoption there are three common characteristics: exploring commonalities and variability, architecture-centric development, and two-tiered organization in which one part develops the reusable assets and the other develops products using the assets [9]. Regardless of the reported successful adoption stories, in the literature no reported experiences of using the adoption models can be found. This study fulfils that gap with respect to one adoption model.

2. Research Questions and Study Setup

There are two main research problems considered in this study. The problems are further elaborated into sub-questions, which when answered will solve the main research problems. The research problems with their sub-questions are as follows. (1) How can a software product line be adopted? – How to choose an adoption strategy and model among the existing strategies and models presented in the literature? What are the general evaluation criteria for the selection of the adoption models for an empirical case? (2) How does the chosen adoption model fit for the context of this empirical research? - What are the experiences of using the adoption model? Are there any missing characteristics in the model, which would have been essential in this particular context?

The research performed consists of two parts: a literature review (first research question) and an empirical study (second research question). Based on the literature, the general evaluation criteria are identified to evaluate the adoption models. By evaluating the adoption models, the most suitable model for the needs of the target organization the research will be selected. The adoption is then applied in an empirical case study performed in a multimedia unit of a global telecommunication company.

3. State of the Art of Software Product Line Engineering and Adoption

Principles of software product line engineering are presented to get a common understanding of the approach and its terminology. Thereafter, different factors impeding the successful adoption are described. Different adoption strategies are

introduced to illustrate ways a software product line can be adopted. At the end the different adoption models according to the literature are presented with their common phases and development parties. In addition, general evaluation criteria for the selection of the adoption model are identified and the adoption models are evaluated to find the most suitable one to be used in the empirical part of this research.

3.1 Software Product Line Engineering

Software product line engineering is based on the idea that software systems in a particular domain share more common characteristics than uniqueness and those systems are built repeatedly releasing product variants by adding new features [1]. Therefore, the scope of the product line is defined so, that the products involved in it have a high degree of common characteristics and the implementation of a component is shared over multiple products [2, 9]. Generally, approximately upfront investments of three or four products are required to have return on those investments [9, 16, and 17]. Nevertheless, using an incremental transition approach with a large legacy code base in a large organization, it is possible to adopt a software product line without a large upfront investment and without disrupting ongoing product schedules [18]. Important issues related to software product line engineering are the definition of scope [19], and the consideration of variability and commonalities [20, 21, and 22].

Architectures have a key role in software product line engineering [9, 19, 23, 24, and 25]. There are three kinds of architectures in the context of product lines: platform architecture, product line architecture, and product-specific architecture. Platform architecture is used to build reusable assets within a platform and it focuses on the internal structure of the platform [26]. A software platform is a set of software subsystems and interfaces that form a common structure from which a set of derivative products can be efficiently developed and produced [15]. A platform is the basis of the product line and, in many cases, is built from components evolving through the lifecycle of the product line. If developers cannot obtain the assets they need from the platform itself, they must develop them. Afterwards, the new, single-product assets might be integrated into the platform. [23].

To achieve the benefits of software product line engineering, an organization has first to adopt the approach. The adoption is a major change process in the organization affecting different groups in the organization [2]. According to Bosch, the different alternatives of adoption should be understood and evaluated, rather than blindly following a standard model [27]. The adoption itself starts with an assessment of the current state [7]. Therefore, it is essential to understand different challenges, strategies and models related to software product line adoption.

3.2 Software Product Line Adoption

Both organizational and technical skills are the key for a product line introduction in existing domains [28]. Challenges related to technical aspects are wrong or incomplete requirements for the platform and wrong platform architecture [26]. In addition, if there is lack of either in architecture focus or architecture talent, an

otherwise promising product line effort can be killed [19]. Software product line adoption affects employees' roles and responsibilities. When an organization learns to operate in a new mode, it is usually not achieved without problems [2]. There are resistances within the adopting organization, which can affect the success of the product line adoption [26].

When moving from conventional software development towards software product line engineering, a selected adoption strategy defines how much investments are needed in the beginning of the adoption and what the development time of the products is. During the adoption, the change effort needed is usually underestimated and timetables are often defined to be too tight [8, 26]. In addition, this is challenging, as normally resources have to be shifted from existing projects, and those rarely have resources to spare [7]. Organizations are typically hesitant to invest in changes if they do not have obvious, short-term Return on Investment (ROI) [2].

One of the most essential issues to take into consideration during the adoption is management commitment. Without explicit and sufficient management commitment and involvement, product line business practices cannot be influenced upon and successful [3, 19]. In addition, management commitment needs to be long-term [13] and it doesn't depend on the size of an organization [29]. According to Krueger [30], minimally invasive transitions eliminate the adoption barriers. This means that while moving from conventional one system development towards software product line engineering, only minimal disruption of ongoing production schedules is allowed. Minimally invasive transitions have two main techniques. The first technique focuses on exploiting existing systems, in which existing assets, processes, infrastructure, and organizational structures of an organization are carefully assessed to exploit them as much as possible. The second technique concentrates on incremental adoption, in which a small upfront investment creates immediate and incremental return on investment (ROI). In such a case, the returns of the previous incremental step fund the next incremental step, and the organization adopts a software product line not much disrupting the ongoing production. In addition, to lower the adoption barriers, the organization's current strengths and interests should be taken into consideration together with a reasonable speed of change [28].

3.3 Software Product Line Adoption Strategies

There are different strategies with different names on how to adopt software product lines. McGregor et al. [9] present two main types of adoption strategies, which they call heavyweight and lightweight strategies. Krueger [8] discusses about proactive, reactive, and extractive adoption models. Further, according to Schmid and Verlage [10], there are four types of situations when adopting a product line: independent, project-integrating, reengineering-driven, and leveraged. Böckle et al. [7] divide transition strategies into four groups, which are called incremental introduction, incremental investment, pilot first, and big bang. Bosch [6] divides the adoption process to two different approaches, evolutionary and revolutionary, for two different situations depending on are the existing items utilized or not. Although there are many different strategies for adopting a software product line, there are common

characteristics among them. Common to all the mentioned adoption strategies is that the adoption either starts from scratch or exploits existing systems.

The main differences between the two strategies are related to duration of the adoption time and needed upfront investment. In the starting from scratch strategy the adoption time (and thus the development time of one product) is shorter but higher upfront investments are needed than in the latter strategy and returns on investment can only be seen when products are developed and maintained. In addition, the cumulative costs are reduced faster in the starting from scratch strategy than in exploitation of existing products. [9,31]. Starting from scratch strategy is like waterfall approach in conventional software engineering whereas exploiting existing systems refers to incremental software development [31].

There are also differences between the strategies in exploiting commonalities and variability, in architecture development, and in organizational structure. In starting from scratch strategy, the adoption starts from creating assets which satisfy the specifications of the platform architecture. After that, creation of products takes place. In addition, product line architecture is defined completely before delivering first products. When using the starting from scratch strategy, there are particular teams which produce assets such as architecture and components. In exploiting existing systems strategy, assets are created from existing and currently developing products and the product line architecture is not completed when the first products are delivered. In that strategy, organizational structure does not change until the first few products have been delivered. [9].

The choice of the adoption strategy may depend on the situation of an organization and market demand. If the organization can afford to freeze conventional software development while adopting the software product line, it can choose a starting from scratch strategy. On the other hand, that strategy would be good in cases where the organization has additional resources for adoption, or the transition doesn't need to be done quickly. In the cases where the organization has already products, or even a product line, which are worth to utilize, it may choose an exploitation of existing systems strategy. That strategy can also facilitate the adoption barrier of large-scale reuse as the organization can reuse existing items (software, tools, people, organization charts, and processes) to establish a product line [8].

3.4 Software Product Line Adoption Models

The adoption of software product line requires changes in technical, management, organizational, process, and personnel aspects [2, 5, and 7]. Consequently, an adoption model needs to take into consideration these aspects, if not all at least most of them. The adoption models focusing on only certain aspects are not discussed in this research, for example the ones where adoption is based on legacy products [32], architecture [33, 34], organizational structure [27], or separation of concerns [35].

Böckle et al. [7] has introduced a *General Adoption Process Model* for adopting a software product line. It has four main phases focusing on stakeholders, business cases, adoption plan, and launching and institutionalizing. In addition to the main phases, the model includes different factors contributing to the adoption: goals, promotion, and adoption decision.

Software product line adoption requires many decisions which have to be made in the adoption phase by an adopting organization. These decisions concern what components are developed and in which order, how the architecture is harmonized, and how the development teams are organized. For that purpose, *Decision Framework* introduces five decision dimensions: feature selection, architecture harmonization, R&D organization, funding, and shared component scoping [2]. In addition the model contains three stages through which product line adoption typically evolves through: initial adoption, increasing scope, and increasing maturity.

Product Line Software Engineering (PuLSE) methodology has a strong product-centric focus for the conception and deployment of software product lines [36]. It comprises three main elements which are deployment phases, technical components, and support components. The deployment phases involve activities which are needed when adopting and using a product line. There are four different deployment phases: PuLSE initialization, product line infrastructure construction, product line infrastructure usage, and product line infrastructure evolution and management. The purpose of the technical components, the second element of the PuLSE methodology, is to offer technical knowledge needed in all the phases of the product line development. There are six technical components: customizing, scoping, modelling, architecting, instantiating, and evolving and managing. The support components are information packages or guidelines, the purpose of which is to enable a better adoption, evolution, and deployment of the product line and they are used by deployment phase components. There are three support components: project entry points, maturity scale, and organization issues.

Business, Architecture, Process, and Organization (BAPO) model is a four-dimensional evaluation framework which organizations can use for determining the current state of the product family adoption and improvement priorities [37]. The dimensions concern business, architecture, process, and organization. Each dimension can be on five different levels which are defined with different evaluation aspects. For example, in business dimension at reactive level, identity of an organization is implicit (software product line engineering not visible), there is only short-term vision and both objectives and strategic planning are missing.

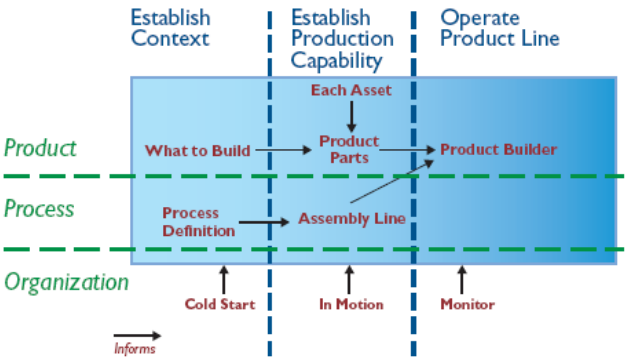

Fig. 1. Adoption Factory [5]

Adoption Factory has, just as a decision framework including three main phases (Establish Context, Establish Production Capability and Operate Product Line) for software product line adoption (Figure 1). Different focus (Product, Process and Organisation) areas are separated by horizontal dashed lines and arrows are the indications of information flows and shift of emphasis among the elements. [5, 38].

4. Conduct of the Study

In the beginning of the study the idea was that the main evaluation criterion for the selection of the adoption model would be derived from the reported experiments of using the models by the adopting organizations. However, no reports were found in the literature describing pros and cons of using the models in the adoption phase. Some of the models had reported experiences in the literature: PuLSE [29, 39, and 40], Adoption Factory [41], 2005, and BAPO [26]. These reports nevertheless did not discuss the applicability of the models.

The empirical research was carried out as a case study. According to Yin, case study is suited for research which is focused on finding answers to "how", "why" or exploratory "what" questions, when the investigator has little control over the events, and when a contemporary phenomenon is investigated in some real-life context [43]. A case study is either single-case or multiple case and the data gathering methods for a case study are surveys, interviews, observation, and use of existing materials. This research focused on a single-case. The empirical data was collected by semi-structured interviews and by analyzing existing materials of the organization.

4.1 Choosing the adoption model

Due to the situation more general evaluation criteria were derived from the literature including: supported adoption strategy, customization, separation of core asset and product development, current state evaluation and guidelines. The *supported adoption strategy* defines to which strategies the model is applicable; starting from scratch, exploiting of existing systems, or both. *Customization* means the ability of an organization to tailor the adoption model for its own needs. *Separation of core asset and product development* defines whether these two development phases are illustrated separately in the adoption model. The *current state evaluation* describes how easy the evaluation is to do in higher level, and may have values easy or not easy. The last evaluation criterion presents, whether *guidelines* for proceeding with the adoption may be followed based on the adoption model. Customization, separation of core asset and product development, and guidelines may have values yes or no.

The adoption models were evaluated according to the defined criteria in order to find the most suitable one for using in the empirical study (Table 1). Based on the

evaluation of the adoption models and the research context, Adoption Factory[1] was selected for the empirical case.

Table 1. Evaluation of the Adoption Models

	Supported Adoption Strategy	Custom-ization	Separation of Core Asset and Product Development	Current State Evaluation	Guidelines
General Adoption Process	Both	Yes	No	Not Easy	No
Decision Framework	Exploiting Existing Systems	No	No	Easy	Yes
PuLSE	Both	Yes	No	Easy	No
Adoption Factory	Both	Yes	Yes	Easy	Yes
BAPO	Both	Yes	No	Not Easy	Yes

4.2 Interviews

The themes for the interviews were selected from the Adoption Factory on the basis of two reasons. As the purpose was to find out current status and future plans of the software product line adoption, the selected themes should cover the model as extensively as possible (but considering the resource limitations of the research) and the interviewees should have knowledge about them. The themes are marked with arrows in Figure 2. The structured questions for the interviews were derived from the selected themes. The questions were partly planned beforehand, but not in very much detail. In addition, there were also questions relating to the gathered experiments which were utilized when defining the adoption guidelines for the target organisation.

Before the interviews, the interviewees were divided to different categories according to different generic development phases of the organization in question. The categories were road-mapping, product management, architecture, and requirements engineering. The reason for these categories was that possible gaps between them, for example in communication, could be found in order to minimize the gaps when proceeding with the adoption. Another reason was to find out if all aspects and steps of maturing market needs for requirements that could be implemented were covered. In the interviews, the themes varied according to which category the interviewee belonged to. Table 2 clarifies the relationships between the themes and the interviewees. As in most of the cases all the selected themes belonging to one sub-pattern were asked from the interviewee, the sub-patterns were used instead of the themes as presented in Table 2.

[1] The Adoption Factory is discussed in some more detail in the empirical section. A detailed description of the Adoption Factory may be found in SEI's web pages [42].

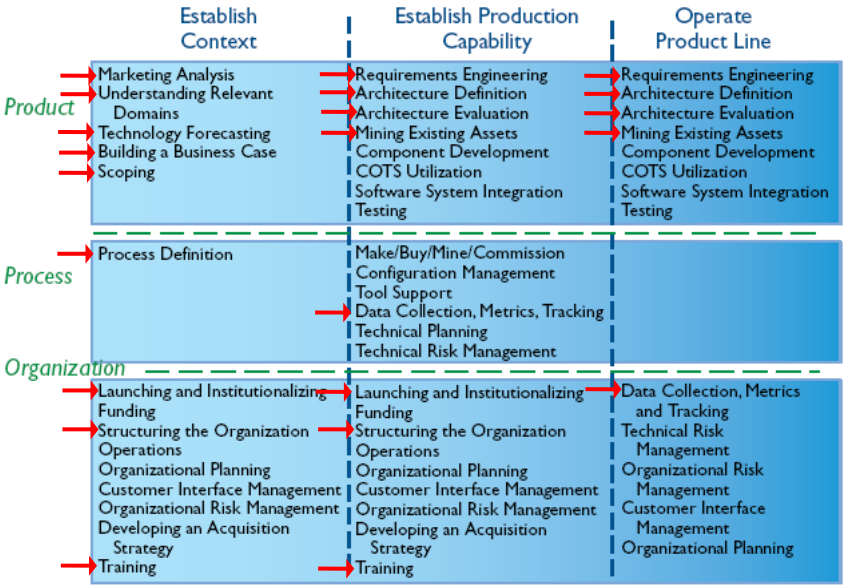

Fig. 2. Selected Themes for the Interviews from the Adoption Factory

Table 2. Summary of the interviewees

Interviewee	Category	Role of the Interviewee	Date
1	road-mapping	Senior Manager, Portfolio Management	7.8.2007
2	road-mapping	Senior Product Manager, Road-mapping	7.8.2007
3	product management	Product Manager	8.8.2007
4	product management	Product Manager	8.8.2007
5	architecture	Engine Product Manager	10.8.2007
6	architecture	Product Chief Architect	13.8.2007
7	requirements engineering	Product Requirement Manager	14.8.2007
8	requirements engineering	SW Technology Manager, Requirements	23.8.2007
9	requirements engineering	SW Requirements Operational Manager	23.8.2007
10	requirements engineering	SW Implementation Operational Manager	23.8.2007

4.3 Data Collection

In addition to the interviews, existing documents were analyzed to clarify the current state and future plans related to the software product line adoption. The

analyzed materials were mainly mentioned during the interview by the interviewee, so the interviews had an open-ended nature. Such material was, for example, a process description of a certain development phase. The existing documents were analyzed after the interviews.

After selecting the adoption model, an e-mail was sent to 10 persons who had participated in the development of the product line and one product involving the product line to inform them about the research. The e-mail consisted of general information of the research and the Adoption Factory together with the purpose of the research. Two days later, a new e-mail was sent to arrange an interview. In that e-mail, there were a list of themes and the topics, which would be covered in every theme: current situation, experiments, and future plans. Therefore, the interviewees could be well-prepared beforehand [43]. No one declined the interview.

Among the interviewees there were two persons from road-mapping, product management, and architecture categories, and four persons from requirements engineering. The roles of the interviewees varied according to which category they belonged to. Overall the interviewees covered the interview themes well. The interviews were conducted in the same order as they are presented in Table 2.

All the interviews were face-to-face interviews with one interviewee at the time. In the beginning of each interview, a short introduction was held to familiarize the interviewees more closely with the research. The introduction consisted of the Adoption Factory, which was gone through more in depth than in the e-mails, research problems (and that the interviews will answer to the second research question), how the research is conducted, and how the results are constituted. The interviewees had a possibility to ask for more details, if necessary.

The interviews themselves lasted for an average of one and a half hours. All the interviews were tape-recorded with a digital voice recorder, so that any of the information they gave would not be wasted and only correct information would be used when analyzing the results. After the interviews, the tape-recordings were transcribed. Later, the data gathered by interviews and by using existing material were read through several times together with the Adoption Factory to form a clear general view to analyze the results more in depth.

4.4 Data Analysis and Results

In this study, the data was analyzed by classifying it according to the used themes. By this, the current situation could be compared to the model, as well as the future plans. With these classifications it is possible to see, if some focus area of the adoption or a part of it is not considered. Together with the categories, the flow between road-mapping, product management, architecture, and requirements engineering could be seen and possible gaps were discovered. Hence, especially conflicts between different categories were noticed, as those would affect the success of the adoption negatively.

The findings were classified according to the same categories, which were used in the categorization of the interviewees. The categories are called road-mapping, product management, architecture, and requirements engineering. In addition, the findings related to several or all categories are discussed in the end of this section.

Table 3 summarizes the findings according to the category they belong to and a possible reason for each finding. Findings and reasons were conducted from the research data (interviews and using of existing material). Adoption Factory was also considered when defining the reasons for the findings.

Table 3. Main findings with their possible reasons

Category	Finding	Reason
Road-mapping	Period requirements described in a too high level of abstraction	Period requirements defined for several product lines
Road-mapping	No commonalities for one product line	Period requirements defined for several product lines
Road-mapping	Period requirements cannot be implemented in a required timeframe	Processing of the period requirements not been defined
Product Management	Each product goes through the period requirements by itself	No commonalities for one product line
Product Management	Documentation requires a lot of effort	Each product team writes its own documentations
Product Management	Documents are not comparable between the products	No common structure for the documents
Product Management	Inefficient communication	No clear roles and responsibilities
Architecture	Architecture definition could not be started before certain decisions related to it were made	Insufficient management commitment
Requirements Engineering	Confusion among stakeholders	No clear roles and responsibilities
Requirements Engineering	Lots of data is collected but it is utilized poorly	No common structure for the metrics
	Product line was established after establishing the products	
	Adoption plan has not been defined	
	No common place for data distribution	
	No training related to software product lines	

Based on the findings, guidelines for correcting and improving the situation in the case organization were formulated. Table 4 describes which aspects to consider when following the guidelines, and what benefits the guidelines would give. Both of these were formulated based on the research data (interviews and using of existing material) as well as on the Adoption Factory model. In the formulation one aspect can give several benefits and one benefit can be a consequence of several aspects. The first two guidelines were named in temporal order as short term new operational mode and long term operational mode. Formulation of the adoption plan may be started immediately while changing to a new short term operational mode. This applies equally to place for data distribution, training, and data collection.

Table 4. How and why to take the guidelines into the daily practices

Guideline	Aspects to Consider	Benefit
Product line is established before the products (new short term operational mode)	• scoping for several product lines • period requirements defined for the product lines • supplier starts to process the period requirements immediately • products of the product line implement the same period requirements • product line is more responsible for documentations	• development of products is more efficient • no need to cancel products development (significant cost savings) • no unrealistic requirements • diminished multiple work loads • utilization of documents is more efficient • data collection is more efficient
Core asset development (new long term operational mode)	• core asset development by exploiting existing systems • attached processes for core assets • establishment of a core asset base	• reuse of core assets • utilization of core assets • development of products is more efficient
Adoption plan	• definition of practices, roles, and responsibilities • definition of different requirement types • definition of usage of period requirements • separation of product line and products	• clear practices, roles, and responsibilities • helps with new operational mode • mitigates adverse effects relating to the changes • utilization of period requirements • communication and cooperation is more efficient • development of products is more efficient • valuable for future product lines
Place for data distribution	• existing data is collected to the same place (e.g. to a web page) • possible pilot project • hierarchical order	• utilization of existing data is more efficient • data can be found more easily • helps in employee networking
Training	• trainings should cover principles of product line engineering, relations between different requirement types, each practice area • other training needs should be clarified	• sharing of knowledge is more efficient • development of products is more efficient
Data collection	• definition of data collectors • definition of review points • separation of product line and products • similar structure for the metrics	• helps in following the software product line adoption • needed changes to refine the product line practices can be identified • efforts for developing products can be seen • decreases duplicate work • metrics are more comparable and utilizable

5. Conclusions

In answering the first research question and two sub-questions the following was found in the literature analysis. There are two basic alternatives, which are called adoption strategies, for adopting software product lines. The first alternative is to do everything from the very beginning and not utilizing any existing systems, which is called a starting from scratch strategy. When using the starting from scratch strategy, the development time of one product is shorter but higher upfront investments are needed than in the other alternative, which is called an exploiting existing systems strategy. In that strategy, existing systems are utilized as much as possible and the cumulative costs are reduced faster than in the starting from scratch strategy. Compared to the conventional software development, the starting from scratch is like a waterfall approach and the exploiting existing systems strategy refers to incremental software development.

Based on the literature review, five evaluation criteria were found for selection of adoption models. As the situation of the organization and market demand as well as the adoption time and needed upfront investments are the aspects, which should take into consideration, when selecting a suitable adoption model, the supported adoption strategy is the first criterion. To clarify whether an adoption model can be adapted to the organizational needs, the customization of the adoption model is the second evaluation criterion. Further, the software product line organization has two different roles: the first role is to develop core assets and the other is to produce products by exploiting the core assets. Due to this, the third evaluation criterion is called the separation of core asset and product development. In addition according to the theory, the adoption should start with a current state evaluation and the possibility for evaluating the current state with the adoption model needs to be considered, when selecting the adoption model. The last evaluation criterion is called guidelines. That means that the adoption model should support the creation of guidelines the purpose of which is to help to keep the adoption in the right track.

In answering the second research question and two sub-questions the empirical part of the study concluded the following. First of all five guidelines were defined be taken into consideration when proceeding with the adoption. The first is to change the operational mode towards software product line engineering. As a short term guideline, the operational mode will be changed to establish the product line before the products involved in it. As a long term guideline, the operational mode is changed to develop core assets and the products are developed based on the core assets. At the same time with the new operational mode, an adoption plan should be created. The purpose of it is to define new practices, roles, and responsibilities needed to adopt software product line. After these, a place for data distribution is needed to utilize existing systems as extensively as possible. In addition, training is needed to ensure that the products of the product lines can be efficiently build. The last guideline is called data collection, which helps to measure if the adoption plan is working and the efforts needed to develop products are available. These three guidelines should be considered in reverse order: data collection should be considered first, then training, and the last, but not least, the place for data distribution should be established.

Secondly the used adoption model, Adoption Factory, was found to be the most suitable one for this research context based on the literature review. The overall

comprehension of the model is that the model was utilizable in the empirical part of the research. The phases and the focus areas of the model enabled the analysis of the organization in question. In addition, the practice areas of the model were clear and understandable when defining the interview themes and questions as well as the guidelines. Based on the model, the current state could be estimated and it was possible to set future guidelines were possible to constitute. The model suited well for the context of the research.

As no reported empirical experiences were found in the literature of using the adoption models, this study fulfils that gap for the Adoption Factory model, although more case studies should be carried out to understand in which context a certain adoption model would suit the best. Two missing characteristics in Adoption Factory were found during the empirical study. First, the model is meant for a pure software product line adoption. It doesn't consider cases where software needs hardware components for its operation and, therefore, totally new practice area could be included in the Establish Context phase for considering architectural aspects of embedded software product lines. Secondly, a new practice area or even an alternative phase could also be added to the Establish Context phase to show how to share the results of the marketing analysis between several product lines.

References

1. Sugumaran, V., Park, S., & Kang, K. C. (2006). Software product line engineering. Communications of the ACM, 49(12), 28-32.
2. Bosch, J. (2004). On the development of software product-family components, Lecture notes in computer science (pp. 146-164). Berlin / Heidelberg: Springer.
3. Birk, A., Heller, G., John, I., Schmid, K., von der Massen, T., Muller, K., et al. (2003). Product line engineering: The state of the practice. Software, IEEE, 20(6), 52-60.
4. Clements, P., & Northrop, L. (2002). Software product lines: Practices and patterns. Boston: Addison-Wesley.
5. Clements, P. C., Jones, L. G., McGregor, J. D., & Northrop, L. M. (2006). Getting there from here: A roadmap for software product line adoption, Communications of the ACM, 49(12), 33-36.
6. Bosch, J. (2002). Maturity and evolution in software product lines: Approaches, artefacts and organization. Software product lines : Second international conference, SPLC 2, San Diego, CA, USA, august 19-22, 2002 (pp. 257-271). Berlin / Heidelberg: Springer.
7. Böckle, G., Munoz, J. B., Knauber, P., Krueger, C. W., Sampaio do Prado Leite, Julio Cesar, van der Linden, Frank, et al. (2002). Adopting and institutionalizing a product line culture. Lecture notes in computer science (pp. 1-8). Berlin / Heidelberg: Springer.
8. Krueger, C. (2002). Eliminating the adoption barrier IEEE Software, 19(4), 29-31
9. McGregor, J. D., Northrop, L. M., Jarrad, S., & Pohl, K. (2002). Initiating software product lines, IEEE Software, 19(4), 24-27
10. Schmid, K., & Verlage, M. (2002). The economic impact of product line adoption and evolution, IEEE Software, 19(4), 50-57
11. Brownsword, L., & Clements, P. C. (1996). A case study in successful product line development, Pittsburgh: Carnegie Mellon University, Software Engineering Institute.
12. Donohoe, P. (2000). Software product lines: Experience and research directions. Berlin/Heidelberg: Springer.

13. Jaaksi, A. (2002). Developing mobile browsers in a product line, Software, IEEE, 19(4), 73-80
14. Kiesgen, T., & Verlage, M. (2005). Five years of product line engineering in a small company. Proceedings of the 27th International Conference on Software Engineering, 534-543.
15. Meyer, M. H., & Lehnerd, A. P. (1997). The power of product platforms: Building value and cost leadership. New York, N.Y.: Free Press.
16. Pohl, K., Böckle, G., & van der Linden, F. (2005). Software product line engineering: Foundations, principles, and techniques. Berlin: Springer.
17. Weiss, D. M., & Lai, C. T. R. (1999). Software product-line engineering: A family-based software development process. Reading (MA): Addison-Wesley.
18. Hetrick, W. A., Krueger, C. W., & Moore, J. G. (2006). Incremental return on incremental investment: Engenio's transition to software product line practice. Conference on Object Oriented Programming Systems Languages and Applications, 798-804.
19. Northrop, L. M. (2002). SEI's software product line tenets, IEEE Software, 19(4), 32-40
20. Bosch, J., Florijn, G., Greefhorst, D., Kuusela, J., Obbink, H., & Pohl, K. (2001). Variability issues in software product lines. Software product family engineering: 4th international workshop, PFE 2002, Bilbao, Spain, October 3-5, 2001 (pp. 303-338). Berlin / Heidelberg: Springer.
21. Jaring, M., & Bosch, J. (2002). Representing variability in software product lines: A case study, Proceedings of software product lines : Second international conference, SPLC 2, San Diego, CA, USA, august 19-22, 2002 (pp. 19-22). Berlin / Heidelberg: Springer.
22. Coplien, J., Hoffman, D., & Weiss, D. (1998). Commonality and variability in software engineering, IEEE Software, 15(6), 37-45
23. van der Linden, F. (2002). Software product families in Europe: The esaps & cafe projects. IEEE Software, 19(4), 41-49.
24. Bosch, J. (2000). Design and use of software architectures: Adopting and evolving a product-line approach. ACM Press/Addison-Wesley Publishing Co. New York, NY, USA.
25. Mohagheghi, P., & Conradi, R. (2004). Different aspects of product family adoption, Software product family engineering (pp. 459-464) Berlin/Heidelberg: Springer.
26. Wijnstra, J. G. (2002). Critical factors for a successful platform-based product family approach, Lecture Notes in Computer Science, 2379/2002, 1611-3349, Proceedings of Software Product Lines: Second International Conference, SPLC 2, San Diego, CA, USA, August 19-22, 2002. Springer Berlin / Heidelberg
27. Bosch, J. (2001). Software product lines: Organizational alternatives. Proceedings of the 23rd International Conference on Software Engineering, 91-100
28. Stoermer, C., & Roeddiger, M. (2002). Introducing product lines in small embedded systems, Software product family engineering: 4th international workshop, PFE 2002, Bilbao, Spain, October 3-5, 2001. Revised papers (pp. 101-112) Springer
29. Knauber, P., Muthig, D., Schmid, K., & Wide, T. (2000). Applying product line concepts in small and medium-sized companies, IEEE Software, 17(5), 88-95
30. Krueger, C. W. (2006). New methods in software product line practice. Communications of the ACM, 49(12), 37-40.
31. Frakes, W. B., & Kang, K. (2005). Software reuse research: Status and future. IEEE Transactions on Software Engineering, 31(7), 529-536.
32. Simon, D., & Eisenbarth, T. (2002). Evolutionary introduction of software product lines, Lecture Notes in Computer Science, 2379/2002, 1611-3349, Springer Berlin / Heidelberg. Proceedings of Software Product Lines: Second International Conference, SPLC 2, San Diego, CA, USA, August 19-22, 2002.
33. Myllymäki, T., Koskimies, K., & Mikkonen, T. (2002). Structuring product-lines: A layered architectural style

34. Thiel, S. (2002). On the definition of a framework for an architecting process supporting product family development, Software product family engineering: 4th international workshop, PFE 2002, Bilbao, Spain, October 3-5, 2001. Revised papers (pp. 3-47) Berlin/Heidelberg: Springer.
35. Krueger, C. W. (2002). Easing the transition to software mass customization, Proceedings of the Distal Seminar No.01161: Product Family Development
36. Bayer, J., Flege, O., Knauber, P., Laqua, R., Muthig, D., Schmid, K., et al. (1999). PuLSE: A methodology to develop software product lines. SSR '99: Proceedings of the 1999 Symposium on Software Reusability, Los Angeles, California, United States. 122-131
37. van der Linden, Frank, Bosch, J., Kamsties, E., Känsälä, K., & Obbink, H. (2004), Software product family evaluation, Lecture notes in computer science (pp. 110-129). Berlin / Heidelberg: Springer.
38. Northrop, L. M. (2004). Software product line adoption roadmap, Pittsburgh: Carnegie Mellon University, Software Engineering Institute.
39. Kolb, R., Muthig, D., Patzke, T., & Yamauchi, K. (2005). A case study in refactoring a legacy component for reuse in a product line, Proceedings of the 21st IEEE International Conference on Software Maintenance, 2005, 369-378
40. Schmid, K., John, I., Kolb, R., & Meier, G. (2005). Introducing the PuLSE approach to an embedded system population at Testo AG, Proceedings of the 27th international conference on Software engineering, 544-552
41. Donohoe, P., Jones, L., & Northrop, L. (2005). Examining product line readiness: Experiences with the SEI product line technical probe. Proceedings of the 9th International Software Product Line Conference,
42. Northrop, L. M., & Clements, P. C. (2007). A framework for software product line practice, Retrieved 05/03, 2007, from http://www.sei.cmu.edu/productlines/framework.html
43. Yin, R. K. (2003). Case study research: Design and methods (3rd ed.), Thousand Oaks, California: Sage Publications, corp.