Three Case Studies on Initiating Product Lines: Enablers and Obstacles

Andreas Birk

sd&m AG, Industriestraße 5, D-70565 Stuttgart, Germany, e-mail: andreas.birk@sdm.de

Abstract

This position paper contains three case reports of projects in which software product line engineering or some key principles of it were applied. From these cases, conclusions about technical and organizational prerequisites for product line initiation are derived.

1 Introduction

Software product lines (SPL) promise high productivity and quality gains. But adoption of SPL practices is not easy: Still relatively few people know the approach and trust into it, upfront investments are required, and certain development process capabilities are mandated that are not always among the standard practices of a software organization (e.g., systematic requirements management and advanced configuration management). This paper tries to shed light on the factors that can support or hinder the implementation of SPL in software projects and organizations.

Clements and Northrop [1] define a software product line (SPL) as 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.

What technological and organizational obstacles must be overcome for a project or company to initiate and implement SPL? This paper addresses this question from the perspective of a software house, sd&m, that develops custom software solutions. The question is addressed through three case studies: Two cases are about projects of sd&m where SPL was a possible development model but could only partially be implemented. The third case describes a company-wide initiative of sd&m that has established a generic SPL infrastructure. The infrastructure enables the development of custom information systems for different clients as variants of one overall SPL. For each of the three cases, the paper presents observations on enablers and obstacles of SPL. A final conclusions section summarizes the observations.

2 Case 1: UI Framework Components

The first case is a project that developed user interface (UI) framework components for a new generation of technical devices (embedded systems). sd&m's client develops and manufactures such devices as an original equipment manufacturer (OEM) for its own customers. New products are usually developed for one pilot customer and its specific requirements. Later, the product will be extended and modified to suit the needs of additional customers.

Traditionally, variability of technical features across device variants for different customers used to be quite low. Software development was reuse-oriented but would not qualify as SPL engineering. With the advent of graphical UIs, a new key differentiating factor among the customer-specific product variants has arisen. Every customer now wants its own specific graphical appearance and user interaction, imposing new and highly complex requirements on software development.

sd&m became engaged when a new product generation had to be developed and the client did not want to cover all development efforts itself. sd&m got contracts for some sub-projects, among them development of UI framework components. Project responsibility of this sub-project was with sd&m, in close collaboration with the client's overall project management and related sub-projects run by the client or sd&m.

The new system was developed for a new domain-specific platform that included operation system, middleware infrastructure, and UI framework. The existing UI framework offered various useful features (e.g., abstraction from device drivers and flexible changes between graphical styles). But soon it became clear that the existing UI framework could not satisfy several important requirements of the client and its customers. For instance, there was no support for animated graphical effects. So it was decided to set up a sub-project to develop a specific set of own UI components. They had to be integrated with the existing UI framework of the system platform. Application developers would later use them to build user interfaces of different customer-specific product variants. The client viewed the collection of new UI framework components as a core asset that ensures flexible and rapid implementation of customer requirements.

SPL approaches were considered in the project in two respects: First, the UI of the final product (i.e., the embedded system) was a product line. The UI framework was the core asset of this SPL. Second, the collection of individual UI framework components was developed as a small, project-internal product line: It soon became clear to the development team that each UI framework component fell into one of two technical categories, depending on how it had to interact with and comply to parts of the existing system platform (e.g., registration at the UI manager, implementation of certain interfaces, event logging). So the core assets were the generic components needed for developing UI framework components as instances of each type. The core assets included code templates and associated guidelines. They were developed before starting the implementation of the actual UI framework components. Availability of these core assets helped the project very much to keep within schedule and budget. It also ensured high quality of the components.

Observations

SPL scoping can be based on technology-focused assets. - SPL scoping needs not focus on application-oriented, functional commonalties of a family of software systems only. It can also address technical aspects like communication patterns between components and event logging. In the case presented above, such technical commonalties provided the basis for a SPL of individual UI components (each having different functionality) that structured the development activities within one development cycle (i.e., for developing one variant of the UI framework). This technology-focused SPL has proven highly effective for attaining short-term benefits within the project.

In some situations, an incremental approach to SPL scoping is mandatory. - In the described project setting, a pro-active scoping effort of the overall SPL of user interfaces for embedded devices would not have been feasible. Rather, incremental scoping was mandatory. First, it was not possible to access other clients than the current pilot clients. Second, the range of possible requirements that the SPL should fulfil was so wide that it could not have been implemented while also fulfilling the hard runtime and memory utilization requirements of the embedded device.

Need for supporting awareness-creation of SPL engineering at clients and project sponsors. - Project sponsors of the described project were very much interested into reuse. However, they were not aware of SPL engineering, its potential, and prerequisites for its successful implementation. Project schedules hindered project management to start comprehensive SPL awareness creation initiatives. In this situation it would have been a clear advantage for the dissemination of SPL engineering, if SPL dissemination material and business case descriptions could have been readily available from the public domain. Despite our attempts, we have not been able to identify such material.

3 Case 2: National Variants of Corporate-Wide Information System

The second case appears from the outset like a typical SPL example: The development of an information system that had to be instantiated in about a dozen national variants in different countries. It had to have a core that standardizes certain business processes of the national subsidiaries of an international corporation. But every national subsidiary also required several country-specific additions and variations (e.g., considering specific national laws, specific business processes that were

rooted deep in the organization of the subsidiary, and constraints from neighbor systems with which the new system had to interact).

This case description observes the project in a quite early stage from initial project set up to completion of the business process modeling for the pilot client (i.e., one of the national subsidiaries). However, already at this early stage, several factors could be identified that have impacted the possibilities for SPL initiation.

Sponsor of the project was a central business department of the international corporation. Its interest was to standardize and unify the IT solutions existing in the various national subsidiaries. Each subsidiary, however, operated as a widely autonomous entity. The subsidiaries were interested in standardization, but each subsidiary also had its own specific objectives and requirements for IT support. Some subsidiaries had just recently set new systems productive, while others were about planning the substitution of legacy applications. Initiatives about IT standardization had to be triggered by the central business department but decided in consensus across all major subsidiaries. Before drawing IT-related decisions, central department and each subsidiary had to consult their associated IT departments, which sometimes had their own specific agenda. For this reason, decision making about requirements and architecture of the new IT system required quite complex negotiations.

When sd&m was engaged for the project, it was decided that a new system had to be built. Several further details were still to be clarified. So the project first had to scope the system and identify requirements. One national subsidiary was willing to act as pilot client for the first instance of the new system. They were about planning a new round of modifications to their existing system and did now stop this in favor of the new system to be developed. Development of this first system instance was roughly estimated to last 18 months with a peak team size of 15 persons. One other national subsidiary agreed to serve as partner for analysis of commonalties and differences once the pilot client's requirements were defined. The other subsidiaries could be accessed through monthly committee meetings in which they could be asked to introduce specific aspects of their present IT systems and business processes. Any closer collaboration with the subsidiaries was infeasible at this stage of the project. SPL concepts were not known to any of the client organizations and did not find their way into the project contract.

It was clear to the sd&m team that SPL engineering was the key model to system development in this case. However, due to complexity of the development task and limited access to client sites, a comprehensive scoping activity across several subsidiaries was impossible. Instead, the project focused on in-depth analysis of the pilot client. Once the pilot client's requirements and preferred system architecture were clear, a concurrent analysis of commonalties and differences to other subsidiaries had to be set out. Although the initial project setting appeared to request a by-the-book SPL approach including pro-active SPL scoping, organizational constraints made this impossible. The project applied SPL principles wherever possible. But it was not perceived useful to introduce the client organizations explicitly to SPL concepts, mainly for the reason that their interest was on business concerns and they were not very familiar with software engineering and SPL concepts. However, it would have been beneficial if it had been possible to discuss the benefits of SPL to the project in more detail. The following observations summarize the experiences from this project case for the application of SPL in such settings.

Observations

An incremental approach to SPL scoping can be inevitable in some situations. - Whether to perform SPL scoping pro-actively or incrementally is often a question of personal preferences or one's willingness of taking risks in the expectation of great opportunities. In the case addressed here, there was no choice: SPL scoping had to be performed in an incremental fashion. Identification of requirements and comparative analysis of the situations at several subsidiaries were too complex and expensive as if it could have been performed as a purely pro-active effort. Also the communication and decision processes required to get a pro-active SPL effort approved by the sponsors would have been too complex. The only feasible alternative was an incremental approach, starting with one pilot client and gradually evolving the SPL core as additional clients are included.

Systematic requirements engineering is a prerequisite for incremental SPL scoping. - Incremental SPL scoping places particular importance to advanced requirements management and engineering practices. Some SPL core requirements from the pilot customer must later be revised and the SPL core must be modified. This is only possible, when it is always clear what requirements are associated with which parts and features of the system.

Awareness material on SPL engineering, its principles, and its benefits should be readily available for attaining the buy-in of important stakeholders. - The adoption of new technology like SPL requires that awareness of the new approach is created within an organization, and that technical competence can be built among all stakeholders. Decision makers want information about costs and benefits of the approach, and technical staff demands information about software engineering practices, tool support etc. In the case described above, like in case one, the sponsor organization wasn't well aware of SPL engineering and its advantages. But the software project organization couldn't provide the awareness material at the right point in time, due to lack of resources. With appropriate information material readily available, for instance from the public domain, it could well have been possible to root SPL engineering deeper in the project. Since this might happen in many other projects as well, it could be worth starting a SPL community effort for creation of such information material.

4 Case 3: sd&m Standard Architecture for Information Systems

The third case describes sd&m's initiative for defining a standard architecture of information systems that eases the development of custom information systems. This apparent contradiction of developing a *standard* architecture for *custom* information systems is actually a powerful application of SPL principles to a family of functionally diverse software systems. The commonalties between the systems are with their technical architectures: Many information systems have the same three tiers architecture, use the same few database products, application servers, and client technologies, and have similar technical links to neighbor systems. This is enough commonalty - besides all functional differences between banking applications, production planning systems, etc. - to qualify for defining a SPL based on technical systems characteristics.

sd&m's standard architecture for information systems, called *QUASAR* (*Quality Software Architecture*) has been introduced by Denert and Siedersleben [2]. It is built on the distinction of technical software components from application-dependent, functional ones. Each software component should be designed so that it addresses either technical or application-oriented concerns[1]. This makes it possible to define a generic systems architecture, which enables reuse across functionally diverse custom software solutions.

The QUASAR set of reusable assets consists of the generic architecture, software frameworks and reusable components, as well as various methods and processes. The architecture defines standardized interfaces between technical components. For many interfaces, several alternative implementations for different system platforms are available. The strong standardization of interfaces reduces dependencies between components and fosters reusability. This is contrary to conventional frameworks that tend to have extensive interfaces, which increase dependencies and reduce reusability. The methods and development processes associated with QUASAR foster the separation of technical and application-dependent concerns early in the development process. They map system requirements, specification, and system architecture onto the generic QUASAR architecture. This provides the ground for application of frameworks and generic components, and is the basis for well-structured and modular high-quality system architectures.

[1] There are exceptions from this principle, of course. Some software is neither technical nor application-dependent. In other cases, both aspects can not be separated. For details see [2]. However, there must be clear rules for such exceptions. Then it is possible to define a stringent generic systems architecture that is highly reuse-enabling.

The entire company staff is educated in the new standard architecture. Therefore, a series of lectures has been set up, every software engineer receives hardcopies of technical white papers on the approach, and the contents of the entire internal education program are aligned with QUASAR. Knowledge brokers support application of QUASAR, and a staff of technical experts act as internal consultants for the deployment of the software frameworks and generic components in projects. Project reviews, which are performed regularly as part of the company's quality management system, also check whether the projects take full benefit from the core assets provided by QUASAR.

QUASAR and its various supporting measures have been defined in a series of pro-active efforts of sd&m's research division in collaboration with senior staff from all over the company. So the approach integrates new architectural principles with proven and well-established development practice from a variety of projects. Benefits of QUASAR include faster development cycles, faster and highly accurate development of project offers, increased reuse rates (of software components, domain and technical expertise, as well as the various work products of projects), higher staff qualification, and the gradual establishment of a SPL- and reuse-based development approach throughout the company.

Observations

SPL scoping based on technical aspects of software architectures is as least as powerful as scoping of application-oriented functionality. - QUASAR shows that SPL scoping needs not be limited to functional aspects of a family of software systems. Also technical aspects of system architecture can be a source of commonalties, providing the basis of a highly useful core assets. Moreover, for sd&m's business (i.e., development of custom information systems), functionality-based SPL scoping is virtually the only way to benefit from SPL principles and make effective reuse happen.

Pro-active SPL development can be performed in parallel with conventional development; gradual migration to SPL-driven development can be reasonable. - The QUASAR case shows that SPL initiation can be performed in parallel with conventional development. This seems to be applicable for every organization that follows a project-based development model. Projects are free to decide when they find it most suitable to switch to SPL development. They also can do it in a gradual manner, adopting only parts of the SPL assets at one point in time. A prerequisite is that the basic systems architecture is compatible with the core assets' architecture. Such architectural standardization might be much easier to achieve for the technical aspects of an architecture than for the application-oriented, functional ones. The reason is that technical domains are generally more standardized (i.e., de-facto industry standards etc.) than application domains.

Active awareness creation and dissemination of SPL principles are critical for success; at some point in time, demand for SPL engineering can become self-sustained. - The QUASAR approach is subject to comprehensive company-internal dissemination activities. These activities focus both on creating general awareness and developing technical competence for applying the approach. As a result, nearly every software engineer at sd&m has now basic knowledge of QUASAR. Project managers are increasingly often experiencing benefits of the approach when they write project offers or when concerned with technical project issues. This further fuels acceptance and actual application of the approach. It seems that the initial dissemination efforts have created a basic interest into the approach, which now unfolds its own dynamics and raises new demands for further information and support.

SPL engineering must be anchored in a company's development processes. - Much of QUASAR's attractiveness is supported by the fact that it is not just an architectural framework with associated software components, but that it is also rooted in the company's development processes and recommended good practices. Guidelines and document templates have been developed for the early phases of software projects, which help structuring software systems so that they comply with the QUASAR architecture. So, for achieving smooth SPL adoption, the architectural SPL assets should always be packaged together with associated guidelines, development processes, and support tools.

5 Conclusions

This section derives conclusions from the observations made in the three case studies. Observations focus on the key question of the paper, namely, how observed obstacles of SPL initiation can be overcome. The observations are compared and generalized as permitted by the observed evidence and their degree of dependency from specific situational characteristics of the cases described above.

Consider to build the SPL on technology-focused core assets instead of application-oriented, functional core assets. - This conclusion is supported from the experiences made in cases one and three. Technology-focused core assets often allow for higher degrees of standardization, and SPL scoping and core asset definition can be performed with less user involvement, which makes scoping less expensive. Technology-focused core assets also allow for faster return of investment (cf. case one) and can more easily be rooted deeply into the project's or organization's development practices (cf. case three). These observations are also supported from another case study at sd&m [5], which showed that standardization of application-oriented requirements for a client-side GUI framework was much harder than standardization of technology-related requirements.

Investigate whether incremental SPL scoping will have advantages over pro-active SPL scoping. - There is reported evidence that pro-active SPL scoping can create high productivity and quality gains [1] [3]. However, case one and two show also that incremental SPL scoping can be mandatory in some cases. Reasons can be limited access to customers (case two), difficulties in assessing the relevance of future requirements (case one), complexity of the task and/or project environment (case two), or limited availability of financial or personnel resources (case one). Case one has also shown that incremental SPL scoping has advantages for technology-focused core asset development in the context of new technology. It can support the learning curves involved in mastering new technology and avoid costly late modifications of the core assets.

When initiating a SPL initiative, incremental SPL scoping should be considered an alternative to pro-active SPL scoping. Important prerequisites for incremental SPL scoping are mature requirements engineering and change management practices, and the ability of refactoring the core assets. Also the SPL scoping method itself should be suitable. Schmid [4] argues that scoping methods sometimes lack the ability of integrating new, project-specific objectives, and that they are not really replicable. Integration of specific objectives and replicability are important prerequisites for incremental scoping.

Define usage support for core assets so that SPL engineering is deeply rooted in the development practices of the project or organization. - Cases one and three have shown that core asset should be rooted in the project's or organization's development practices. Guidelines that explain how to use the core assets are not enough. Instead, development practices should be defined or changed so that use of core assets becomes the default development approach. The reported cases also show that it is well possible to get new practices adopted by the staff. Once an new development approach has obvious advantages and is supported by useful guidelines and tools, rapid adoption of the new practices should not really be a problem. Both case examples also suggest that the investments for developing such guidelines and tools can pay off quite easily.

Place strong emphasis on SPL awareness creation. - Case three has demonstrated that it is quite easily possible to create high awareness of SPL engineering throughout a development organization. Prerequisite is the development or availability of appropriate awareness and dissemination material. Cases one and two have shown that possibilities for SPL awareness creation among project sponsors can be limited. One limiting factor is the availability of ready-to-use information material (e.g., slide presentations and white papers). It would be desirable that the SPL community starts an effort for developing such material and making it available in the public domain. This could foster the further dissemination and adoption of SPL engineering throughout the industry.

In general, SPL awareness creation is particularly important when the core assets can be useful for a large number of projects or a large community of software engineers. Like the adoption of SPL-based development practices, also the initial creation of interest into SPL might never be a big problem when the SPL approach and core assets are well-designed and their usefulness is obvious. Project sponsors

as well as software engineers want their project be a success. They are generally open to every approach that - like SPL - helps making for project success.

6 References

[1] Clements, P., Northrop, L.: Software Product Lines: Practices and Patterns. Addison-Wesley, Boston, MA (2002)

[2] Denert, E., Siedersleben, J.: Wie baut man Informationssysteme? - Überlegungen zur Standardarchitektur (in German). Informatik Spektrum, pp. 247-257 (2000)

[3] McGregor, J.D., Northrop, L.M., Jarrad, S., Pohl, K. (Eds.): Initiating Software Product Lines. IEEE Software, 19 (4), July (2002)

[4] Schmid, K.: A Comprehensive Product Line Scoping Approach and Its Validation. In: Proc. 24th Int'l Conf. Software Eng. (ICSE'02), ACM Press, New York (2002)

[5] Stützle, R.: Wiederverwendung ohne Mythos: Empirisch fundierte Leitlinien für die Entwicklung wiederverwendbarer Software. Dissertation, Technical University Munich (2002)