Variability in Standard Software Products

Introducing Software Product Line Engineering to the Insurance Industry

Alfred Bröckers

1 Introduction

The development of industrial goods and consumer products has become remarkably mature over the past century. The availability of large numbers of different products has driven their widespread use. The large number of variants was not achieved by developing each of them individually. Moreover, many goods were and are developed in a way that allows for easily adapting them for different purposes. For example, in the automotive industry, different kinds of trucks are developed serving multiple purposes. There are trucks for long-distance transportation, others deliver ready-mixed concrete to construction sites, etc. Despite these different purposes, the commonalities in their design outweigh by far. Therefore, the different variants are developed based on a common platform and can even be produced on the same production line.

Over the last two decades, software product line engineering (SPLE) emerged as an area of software engineering that follows the same pattern that traditional industries adopted so successfully. However, until now, this promising approach has not spread across all industries that are dominated by—or at least driven by—the use of software systems. This applies especially to the insurance industry: Over the last decades, developing individual software systems or tailoring existing software solutions was the only option for insurance companies. However, competition in the insurance industry and tight IT budgets as a consequence have raised the demand for standard software products over the past few years. To cope with a significant diversity of insurance companies, variability is required from such software products. SPLE promises to provide this variability.

A. Bröckers (✉)
adesso insurance solutions GmbH, Dortmund, Germany
e-mail: broeckers@adesso.de

V. Gruhn, R. Striemer (eds.), *The Essence of Software Engineering*,
https://doi.org/10.1007/978-3-319-73897-0_6

The fact that in the insurance sector the knowledge required to define a platform for a product line is distributed across insurance companies and software vendors, the size of the investment, and the speed of innovation makes it difficult for a vendor to introduce product lines following conventional transition strategies. This chapter proposes a transition strategy that fits to the specific situation in the insurance industry. It might be transferred to other industry sectors which exhibit similar conditions. This strategy enables small- to medium-sized software vendor organizations to base their business on software product line engineering.

The next section introduces software product line engineering as far as needed for this chapter. The following section explains why software product lines are a promising approach to establishing software products for the key processes of the insurance industry and why in this domain traditional introduction strategies most likely will not suit the current practice of software development. Subsequently, the chapter introduces the extended pilot project strategy that overcomes the shortcomings and discusses the pros and cons of the approach.

2 Software Product Line Engineering

Software product line engineering [1] is an area of software engineering. It follows the pattern that traditional industries adopted to provide consumers from a single domain with a large variety of products. According to this approach, development for a software product is based on a software platform that is common to a whole software product line. Deriving a software product from such a platform consists of customizing and extending the platform to the specific needs of the target group (individual customer or group of customers that share the same requirements). Up to this point, this sounds familiar from traditional approaches to adopt existing software solutions and customize them. Characteristic to software product line approach is that the platform is built for this purpose specifically: The platform already defines the possible range of variability of the product line. It accounts for the aspects that can vary across the products which can be derived. The set of variable aspects must be considered very carefully. Missing variability may limit the product range and exclude relevant products from the product line. On the other hand, accounting for variability that does not really vary across the product line raises development and maintenance cost for the platform and the derived products alike.

In software product line engineering, the overall development process consists of *domain engineering* for developing and maintaining the platform for a product line and *application engineering* that deals with deriving specific software products from a platform [1].

Domain Engineering The result of domain engineering is a software platform that is used by application engineering for building software products. The platform contains all the artifacts that are known from traditional development approaches, such as requirements and design documents, code, as well as test artifacts. In traditional "build-and-customize" approaches, differences first come into play when building the final application. In software product line engineering, the commonalities and (known or expected) differences across the product line are accounted for completely when building the reusable base unit, the platform. Domain engineering defines the complete variability of the range of software products that can be derived.

The variability of a product line is defined in terms of the so-called *variation points* [1]. A variation point defines an aspect that can vary across the different members of the product line, that is, the software products that can be derived from the platform. The different specific forms for such an aspect are called *variants*. Domain engineering is not limited to identifying the variation point. If the set of variants (or a subset thereof) for a variation point is well known, they can be developed during domain engineering. Different solution patterns can be used for implementing variation points. Examples are:

- Configuration parameters, for example, read from flat files, XML structures, or databases for simple situations
- (Model-based) code generators
- Design patterns for framework development [2]

Application Engineering An application engineering project develops a specific software product required by a single customer or customer group. Each project must decide if the product line fits the problem at hand. This is the case if the commonalities that the platform of the product line implements apply. Furthermore, the project has to find out if the required system can be derived from the platform: For each of the variation points, a suitable variant is selected. The variant might already exist as part of the platform or can be developed as part of the application engineering project.

In traditional software customizing approaches, application projects must deal with the complete complexity of the whole reuse basis. In SPLE, developing a software product or application is reduced to selecting or developing variants for a known set of variation points. That reduces the development and maintenance cost significantly.

The goal of software product line engineering is to provide customers with different software products at significantly reduced cost compared to individually build products. The cost of a software product, derived from a platform, is the sum of the cost of the application engineering project and a portion of the cost of domain engineering for the product line. This portion is roughly the *n*th part of the overall domain engineering cost, where *n* is the number of members of the product line. According to [3], the break-even compared to single systems development is reached between three to four systems.

Software product line engineering has not yet spread across all the software-driven industries. From the 15 product line examples given in [1], only two can be identified to be not from the command and control software or embedded software domains. Only one of them is from the financial industry domain.

3 SPLE in the Insurance Industry

In the public perception, the internet of things (IoT) and the ever-increasing penetration of software into traditional consumer products are the main innovation drivers. Software-related innovations in the service sectors are often not recognized. It is just those sectors in which the importance of software to business success is expected to increase significantly. The internet is only one driver of this development. Thus, the demand for software and standard software products in these sectors and especially in the insurance industry is expected to increase accordingly.

For software vendors, the diversity of products, processes, and IT landscapes in the insurance industry is an obstacle to successfully introducing standard software products. Software product lines promise to provide the variability required to solve this problem. Still, the specific situation of software development within the insurance business makes it difficult to introduce software product line engineering. This section gives a short introduction to the state of the practice of software development for the insurance industry and then describes the obstacles to introducing product lines.

3.1 Current Situation

Providing insurance services to consumers and companies alike has always been a business that is intangible and primarily based on administering data on insurance contracts and claims. With the emergence of computer technology, software automated this business without any need for sensors and actuators beyond classical input/output devices such as computer terminals and printers. Therefore, software has always been a vital part of day-to-day business for many decades in insurance companies.

Today, software systems largely pervade any business process within the industry. For business processes that are not specific to the insurance industry, standard COTS software has become a common practice. Examples are standard systems for accounting, collections/disbursements, or output management. For business processes that are specific and characteristic for the insurance industry (core insurance processes), most companies use software systems that are specifically built for these purposes. Among these, there are systems for contract management, claims processing systems, and sales management systems.

Until only a few years ago the pervasive pattern for an insurance company was to develop these core insurance systems individually for its own purposes. Large numbers of software and business consultants were required for these in-house developments. Typically, after the initial development, these systems went into maintenance for a period of 15–30 years before reaching the end of their life cycle. Over this long timespan, they accumulated more and more functionality. Thus, when replacing such a system, the business functions that have been paid for more than a decade must be provided in a much shorter timeframe to maintain the functional status quo. Furthermore, replacing a software system almost always involves technology that is new to the old maintenance team. Therefore, for most small- to medium-sized insurance companies, replacing an insurance core system with an in-house development is an investment that can hardly be justified.

As a result, the trend goes to buying complete software systems even for the core insurance processes. However, there are many differences between the individual insurance companies. Among these, there are the size of each company, its culture, the target customer group, the channel of distribution, and the products sold. In addition, each system to be replaced is embedded into a larger IT architecture with numerous other systems. Examples are other core insurance systems or support systems such as database management systems. To account for these differences, a software system needs to be customized significantly before it can be introduced.

Having customized and introduced an existing software system, an insurance company has to decide on the maintenance strategy. Establishing an in-house project for maintenance and further development is one option, although the cost advantage compared to in-house development is limited to the introduction time: Even if several companies individually customize the same system, maintenance of the customized systems equals pure in-house developments. Even worse, for a single insurance company, maintaining the customized system might be burdened with the complexity of redundant functionality that is needed elsewhere.

Another option is to outsource further development and maintenance to the vendor or to some IT service provider. The insurance company will expect synergy from further development of the initial product. However, depending on the extent of the customization, additional costs arise: Each new release of the initial system carries the risk that the changes interfere heavily with the individual modifications and extensions made when introducing the system.

3.2 Transition Strategies

Software product line engineering is one way out of the dilemma between the demand for standard software products and the variability inherent to the insurance industry. adesso insurance solutions GmbH is a software vendor that applies this approach to software development and maintenance of its in|sure software suite. The in|sure suite contains software for the core business processes of insurance companies. The way the in|sure suite is divided into software systems follows a pattern that can be found in many insurance companies around the world: For the health insurance segment, it contains systems for managing contracts and a claims management system. Likewise, for the property/casualty insurance segment, there is a contract management and a claims management system. For the life insurance segment, there is a combined system for contract and claims management.

Within the in|sure suite, each of these systems constitutes a product line. There is a defined set of variation points for each system. When introducing an in|sure system to an insurance company, those variation points are used to derive a customer-specific variant. Most of the variation points address the following aspects:

- Insurance products that the company offers to its customers
- Processes that are specific to the insurance company
- Selection of systems that the in|sure system must communicate with

The means that are used to implement variation points (for an in|sure system) include modelling and code generation techniques (e.g., EMF-based insurance product modelling), typical pattern techniques for framework development [2], customer-specific include statements, and simple configuration files.

The goal of applying SPLE to the in|sure software suite is to account for the variability during the initial development and the maintenance phase. That means the variants that a customer has selected or implemented survive a release upgrade of the platform.

The current practice of software development for the insurance industry results in some major obstacles for introducing SPLE as a strategy for developing standard software systems for the core processes of insurance companies. Basically, there are four transition strategies for organizations [4]. Only two of them apply: the big bang strategy and the pilot project strategy.

Big Bang Strategy Specific to this strategy is a domain engineering phase that is carried out prior to the first application project. That means platform development (except for maintenance and further development) is completed before the first software product is derived. Although the availability of a completed, stable platform even for the first application engineering project is highly desirable, it is hard to achieve: For the big bang strategy, the commonalities and the required variability have to be well known and well defined in advance. Even future trends of the industry must be known as well. Furthermore, domain engineering for a complete platform is a large capital investment and—sometimes even worse—an investment in time. For example, for standard software products such as the in|sure systems, time to market is vital for business success. When following the big bang strategy, there is no proof nor evidence of the feasibility of the platform for a very long time.

Pilot Project Strategy In this strategy, the domain engineering is carried out as part of a pilot project. The pilot project can be regarded as the first application engineering project. It develops the platform, defines the variation points, and develops the variants needed for the first software product. A platform for a product line can only emerge from a pilot project, if profound and settled domain knowledge is accounted for when defining the variation points. This knowledge is hardly achievable for a project that focuses on a single product mainly. In addition, there will always be a tendency to sacrifice the domain engineering character of the project to budget and schedule pressures inherent to software development. Following the pilot project strategy, there is no natural counterpart to this effect.

Although both transformation strategies have already proven successful in many organizations, there are three main reasons why applying them will not lead to the intended results.

Availability of Domain Knowledge The quality of the platform, especially the reuse benefits, highly depends on the availability of profound domain knowledge that is available to domain engineering. In domains where software development consists of in-house development mainly, the knowledge is widespread. In the insurance industry, the domain knowledge required to develop a platform for a software product line is spread over many insurance companies and almost as many small- to medium-sized software development consultant companies.

Domain Knowledge of Insurance Companies Within each insurance company, there is a high-level understanding concerning the insurance domain and the commonalities and differences between companies. Otherwise, competition would put them out of business easily. However, the detailed knowledge on the specifics of business processes and insurance products is widespread across the company workforce, including IT departments. Although some of the employees within a company may have cross-company knowledge, for example, from changing their job, it will be outdated most likely and limited to two or three companies.

There are initiatives, driven by own interest or by sector associations, to define common strategies for new challenges, for example, new legal regulations. These initiatives are limited to single challenges. Neither do they cover established business practices nor the future trends that could be part of competitive advantages.

Domain Knowledge of the Software Industry Most insurance companies have developed the software systems for their core business processes in in-house projects. Typically, a considerable number of external consultants are hired for those projects. Often the consultants or their employer consultant companies are specializing in software development for insurance companies. Therefore, a consulting company could be regarded as a valuable source of cross-company knowledge. Although individual consultants may stick to an insurance company, consulting companies usually have several customers (even at the same time). The recent project history dominates the domain knowledge. It consists of the experience accumulated in the heads of the employees mainly; after all, in most cases, the customer owns the intellectual property right in the project results. The consequence of relying on the domain knowledge of a consulting company is most likely an incomplete, distorted picture of the requirements of the whole industry. Often a high staff turnover makes this situation even worse.

Speed of Innovation Some of the current software systems that are used as a basis for traditional buy-and-customize approaches provide a wealth of functionality that has been gathered over many years. Introducing these systems is accompanied with large individual maintenance cost and technological backlog. However, the functionality is the predominant factor when selecting new software systems. To be able to compete with the traditional approaches, a new software product—regardless if developed from scratch or derived from a platform—must contain the complete functionality needed of the targeted insurance company. This challenge is exacerbated by new must-have features, required either from market pressure or by law: Digitalization of business process has been driving the demand for software over the last year and is far from being completed. Completely new kinds of products emerge, often enabled by the possibilities of the internet. Regulatory requirements (e.g., data privacy, EU directives) have been rising over the past few years and are expected to increase.

In summary, the demand for functionality is increasing dramatically. Unlike other domains, which are used as successful examples for software product line engineering, in the insurance industry there is no hardware development (such as for sensors, actuators, machines that must be controlled) that slows down the speed of innovation. Given this situation, it is virtually impossible to complete a full domain engineering phase and a consecutive application engineering project, keeping the pace of new emerging requirements. To a software vendor, the upfront investment is a high risk: The first software product with business functions that are partially even enforced by law might be completed too late.

Innovator's Dilemma Most of the companies that specialize in software development for the insurance industry follow a traditional business model: They charge their customers according to the consulting or development work performed. To this business model, software product line engineering is a disruptive technology. If a company invests significantly in a disruptive technology, it will inevitably face the *Innovator's Dilemma* [5]. In the battle for limited monetary and personnel resources, the established model almost always wins. For the big bang approach, the large upfront investment with an uncertain business success exacerbates this effect.

Software product line engineering is a promising approach to software development for the insurance industry. It allows for introducing standard software products for the core processes of the insurance industry that account for the variability of insurance companies. Unfortunately, known transition strategies that have proven to be successful in other domains do not fit well.

4 The Extended Pilot Project

Over many years, insurance companies have extended their in-house developed software systems with a plethora of business functions. When replacing these systems, most of those companies will not be able to maintain their status quo in time and reasonable budget following the in-house project pattern. As stated before, standard software products are a solution to this problem. Those systems must account for the variability that insurance companies need to compete in their market. Software product line engineering promises to provide this variability. The preceding section showed that the known strategies for introducing product lines do not fit the situation.

As an alternative to these strategies, this section proposes the *extended pilot project* as an approach that is a compromise between a big bang approach, based on the domain knowledge of several insurance companies, and a pilot project for a single company. The approach involves several insurance companies as charter clients. It overcomes the obstacles to introducing software product lines shown in the preceding section, without compromising on the general validity of the product lines to their respective target group.

This first section introduces the basic setup of the extended pilot project and then defines some criteria that should be considered when selecting insurance charter clients. It defines the roles of the software vendor and the charter clients in the extended pilot project approach. Finally, the pros and cons of the approach are discussed.

4.1 The Setup

The fundamental idea of the extended pilot project approach is to develop the platform as part of a first application engineering project. Unlike with the simple pilot project approach, the resulting platform is aimed at more than a single product right from the start. Involving several insurance companies ensures the functional breadth and the variability that is needed for an entire product line (or at least the first release thereof). Figure 1 gives an overview. The setup is as follows:

- For each of the typical systems that support the core business processes of an insurance company, a specific product line is defined. The product line for managing health insurance policies is an example.
- Several insurance companies are involved in developing the platform for the product line. Their involvement is not distributed evenly. First, there is the pilot charter client, or pilot client for short. The platform for the product line and the product are developed as part of a single project: Domain engineering and application engineering are carried out in parallel. In addition to the pilot client, other insurance companies are involved. Each of these charter clients intends to start an application engineering project that derives its own specific software product in a succeeding application engineering project.
- During the pilot project, the additional charter clients monitor to which extent the software product fulfills their own requirements. The aspects in the functional specification of the system, where the requirements for pilot charter client and additional charter clients differ, act as candidates for introducing variability to the platform. Only the variants for the pilot client are developed as part of the pilot project. Other variants are left either to maintenance and further development of the platform or to following application engineering projects.
- As soon as the pilot platform reaches a state that is sufficiently stable, the additional charter clients can start their own application engineering project. They benefit largely from matching the platform against their own demands throughout the pilot project.

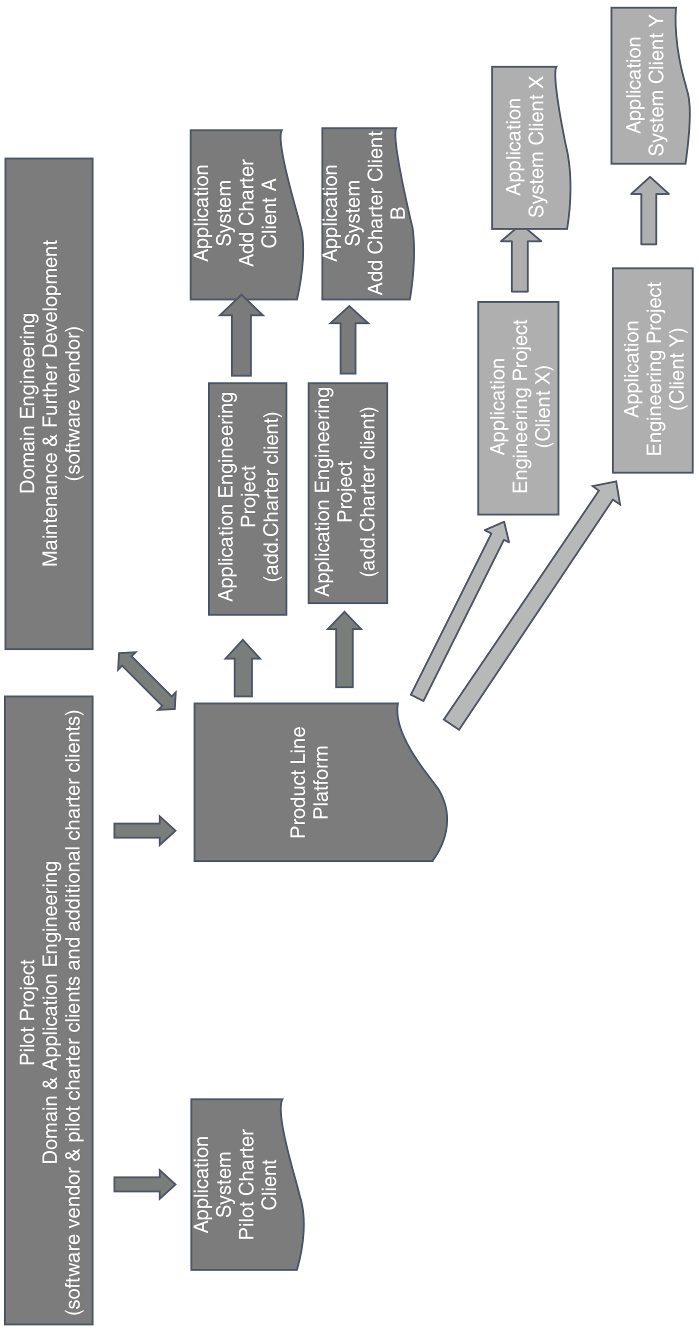
Pilot Project
Domain & Application Engineering
(software vendor & pilot charter clients and additional charter clients)
Domain Engineering
Maintenance & Further Development
(software vendor)
Application System Pilot Charter Client
Product Line Platform
Application Engineering Project (add.Charter client)
Application Engineering Project (add.Charter client)
Application System Add Charter Client A
Application System Add Charter Client B
Application Engineering Project (Client X)
Application System Client X
Application Engineering Project (Client Y)
Application System Client Y

Fig. 1 Overview of the extended pilot project strategy

- After the pilot project has been completed, the platform for the product line enters maintenance state. New customers can derive their own insurance software systems from the platform. Furthermore, from that point on, all application engineering projects including those of the charter clients are equal with respect to maintenance. Maintaining the platform is a pure domain engineering effort carried out by the vendor.

4.2 Selecting Charter Clients

To a software vendor, charter clients shall be considered customers. That means, in the global market of today, the choice is up to the customer and not vice versa. Still, introducing software product lines as part of the business model is a critical endeavor. Therefore, it is important for the vendor to select the charter clients very carefully. The most important selection criteria are a strong motivation, the number of charter clients to include in the extended pilot project, and the contribution of each charter client to the domain coverage of the project.

Motivation The pilot project is of crucial importance for the vendor and for all charter clients. Software development projects almost always face the danger of termination before completion. The reasons are manifold, ranging from internal reasons such as weak project performance to external reasons such as priority changes within the company. While in a traditional project, a termination has only consequences for a single company, in our setup a considerable impact on the entire platform and therefore on all companies involved is inevitable. Thus, each of the charter clients should have a strong motivation to take part in the endeavor. This is especially true for the pilot charter client. Losing a pilot charter client halfway down the road will set the whole endeavor at risk. The pilot client benefits from a much more direct involvement compared to additional charter clients:

- The pilot client has more impact on the definition of the platform, although from a platform point of view this effect is detrimental.
- When defining the platform and the individual variants, the pilot client gets an unbiased alternative view on his business practices.
- The pilot client benefits from the fact that he gets a new system prior to any other client. That might be a competitive advantage.

For an additional charter client, the investment is much smaller than for a pilot charter client. Although the consequences of losing an additional client are much smaller, the impact cannot be ignored. The additional charter client benefits from its involvement:

- While reviewing the platform specifications, he influences the scope and content of the platform. Furthermore, in cooperation with the other charter clients, he defines explicitly the variability that is built into the platform. The impact even goes beyond his own variants.

- In a subsequent application engineering project, an additional client obtains a software system that is customized to his own requirements—a software system that he otherwise would have to acquire at much higher risk and cost.

Number of Charter Clients The number of charter clients has to be balanced carefully. If too few clients are selected, the whole transition strategy degenerates to the pilot project approach with the known effects described before. On the other hand, if too many charter clients are involved, organizational overhead, politics, and discussions on minor details will choke off the progress of platform and product development.

Common sense in the software engineering community is that reuse (and product line engineering is a sophisticated approach to reuse) pays off when a software item is reused three times at least. For software product lines, the break-even is reached at about three derived software products. For this reason, to reduce the economic risk to a manageable size, a software vendor will select a pilot charter client and two charter clients at least. To account for the risk that a charter client cancels its involvement, even more charter clients are desirable.

However, if the number of charter clients exceeds a certain number, some problems outweigh the advantages of adding clients: Quickly obtaining content-related decisions on platform features and on variation points becomes much more difficult. Competition between charter clients and politics step into the decision-making process. Even organizing frequent steering committee meetings becomes virtually impossible. In the insurance industry example, a practical maximum number of charter clients is about five.

Domain Coverage When selecting charter clients, the domain coverage is an important criterion. With respect to software product line engineering, domain coverage can be divided into functional coverage of the platform, variability coverage, and variant coverage. At best, the pilot charter client is a typical representative covering a larger portion of the domain.

Functional coverage is the extent to which the selected charter clients represent the target group of the product line. As an example, Fig. 2 shows how the functional demand can be distributed across the target group and three charter clients.[1] If the functional coverage is too low, the functional scope of the platform may be too narrow to attract further customers.

Variability coverage is the extent to which the selected clients vary. If they do not vary sufficiently, variation points important to the target group will be overlooked. In this case, major rework on the platform will be necessary.

The quality of the platform benefits, if for each variation point the platform development accounts for a representative breadth of variation. Therefore, the charter clients should differ within the boundaries of the target group, requiring substantially different variants. Otherwise, the variation point implementation may

[1]Figure 2 intentionally exaggerates the differences in functional demand. In reality, the differences can be much smaller.

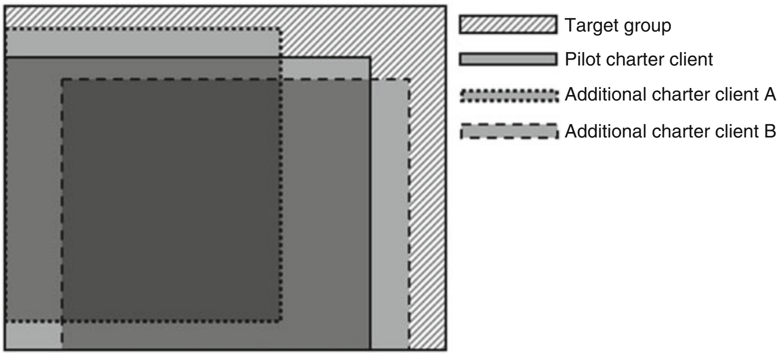

Fig. 2 Distribution of functional demand across the target group and charter clients

be inappropriate for future clients. However, *variant coverage* is hard to recognize in advance.

4.3 Cooperation of Software Vendor and Charter Clients

In our setup, there are several players involved in the pilot engineering project. There are the software vendor, the pilot charter client, and additional charter clients. This paragraph discusses how these players cooperate to provide a platform for a product line and to develop a software product for the pilot charter client. To achieve this, the first application engineering project is set up much like a development project for an individual software system, solely to be used by the pilot chart client: The typical roles of a project such as project manager, requirements engineers, designers, programmers, testers, etc. are staffed using employees of the software vendor and the pilot charter client. The software vendor and the pilot charter client account for the additional cost of domain engineering by providing additional personnel.

Each of the additional charter clients provides two to three proven domain experts taken from its own organization. These experts take part in the functional specification as their main task. This enables them to monitor the specification for completeness and to identify the need for variation with respect to their own organizations. Furthermore, the experts discuss each variation point of the platform and the possible variants with other experts within their own organizations.

Even though the common platform is the result of the pilot project as the first application engineering project, the individual primary interests of the players diverge considerably. Sooner or later the divergence might result in conflicts that set the success at risk. To manage the emergence of these conflicts, a steering committee should be established. This committee consists of top-level decision makers from the software vendor and from the charter clients. It meets on a regular basis and provides for a balance between the interests.

4.4 Pros and Cons

This section discusses the advantages and disadvantages of the introduction strategy described in this chapter: In the insurance industry case, the traditional pilot project strategy would involve a single insurance company. In contrast, the extended pilot project strategy involves several companies. This ensures a functional scope that is much more relevant to the target group and a much higher level of variability.

The big bang introduction entails the risk of supposed demand for variability. Under the pressure to identify sufficient variability, the domain engineering will tend to define more variation points than required. There is no counterpart.

In the extended pilot project strategy, the fact that the platform specification and the variability built into it are validated against the demand of several clients minimizes the risk of supposed variation points: When discussing variation points, the charter clients separate the variability that is really needed from the variability one might think about. The charter clients can even turn supposed variability into common features of the platform. Minimizing the variation leads to an increase in reuse, which in turn reduces the cost of the individual software products.

However, the focus on the demand of the pilot charter client can contain some risks: The extended pilot project provides the pilot charter client with a software system for his organization. Although there is a motivation to develop a platform that a whole product line is based on, the ubiquitous time and budget pressures lead to a tendency to omit aspects that have no relevance to himself. It is the task of the vendor and the additional charter clients to counteract. If conflicts arise, they must be resolved by the steering committee. If this is not possible, the minimum solution is to account for extending the platform later, for example, in the second or third application engineering project or as part of platform maintenance.

5 Summary

Software product line engineering is a promising approach that does not only apply to domains where software products are a well-established concept. Furthermore, it can be applied to domains where—due to a demand for variability—the traditional in-house development project still dominates the development of software. For example, adesso insurance solutions has adopted software product lines to provide software products for the core business processes of insurance companies.

However, software product line engineering is a disruptive technology. To a software vendor, introducing this technology is a long-term investment and a business risk. The current structure of the software development practice, the spread of domain knowledge, the speed of innovation, and the general risks, associated with adopting disruptive technologies, are the major obstacles. Known transition strategies for introducing software product line engineering do not fit this situation well.

The extended pilot project strategy is an approach that accounts for these obstacles explicitly. When following this approach, a pilot project develops the platform for a software product line and derives the first software product. In contrast to other transition strategies, it involves several insurance companies to achieve the functional breadth and the variability needed for the product line.

There are some criteria that the insurance companies taking part in the endeavor have to meet: Each of them must have a strong interest in the product line, given as an explicit plan to derive its own software product. Their individual demand for functionality and variability must contribute to the intended domain coverage of the product line. Finally, the minimum number of companies to be included in the pilot project should account for the risk of a company dropping out. At the same time, the number must be limited to minimize political and competitive effects and to keep organizing feasible.

References

1. Pohl, K., Boeckle, G., van der Linden, G.: Software Product Line Engineering- Foundations, Principles and Techniques. Springer, Berlin (2005)
2. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns – Elements of Reusable Object-Oriented Software. Addison-Wesley, Boston, MA (1994)
3. Weiss, D.M., Lai, C.T.R.: Software Product-Line Engineering – A Family-Based Software Development Process. Addison Wesley, Reading, MA (1999)
4. Boeckle, G., Bermejo, J., Knauber, P., Krueger, C., Leite, J., van der Linden, F., Northrop, L., Stark, M., Weiss, D.: Adopting and instititionalizing a product line culture. In: Proceedings of the 2nd Conference on Product Lines, San Diego, LNCS 2379, pp. 48–59. Springer, Berlin (2002)
5. Christensen, C.M.: The Innovator's Dilemma: When New Technologies Cause Great Firms to Fail. Harward Business School Press, Boston, MA (1997)