

Towards Strategic Reuse in Project-Oriented Environments by Using Software Product Line Practices

Kathrin Berg, Judith Bishop
University of Pretoria
{kberg, jbishop}@cs.up.ac.za

Jay van Zyl
SystemicLogic Research Institute
jay@systemiclogic.com

Abstract

Software product line engineering has become a widely used and popular approach for developing products that are targeted at a specific market. Strategic reuse is a key factor in its success. Large financial organizations, especially in South Africa, cover large domains and thus have huge existing infrastructures running in separate business units. Systems running on these infrastructures have been separately built in each business unit using a project-oriented approach. This implies that a system is built to satisfy a single user's requirements, thus there is a high possibility of duplicate systems occurring across business units of an organization, costing them enormous amounts of money. In order to reduce development and maintenance costs for these organizations, we propose a product line-oriented approach to incorporate strategic reuse into existing infrastructures of large organizations. These concepts have not been applied at a level of granularity that is most beneficial for large organizations. Our approach facilitates the entry of product line practice into large organizations by means of requirements analysis, separation of concerns and feature modelling.

1. Introduction

The software engineering industry is becoming increasingly demanding by requiring software of high quality to be developed in shorter periods of time and within reduced budgets. Strategic reuse [9] is a key factor in achieving these goals. Thus, focus is being shifted from developing single software systems to families of systems or product lines. A system family, according to [7], is “a set of systems sharing enough common properties to be built from a common set of assets”, whereas a product line, being closely related to a system family, incorporates the characteristics of a system family, but is distinguished by mainly focusing on satisfying a given market.

1.1. Project-Oriented vs. Product Line-Oriented Development

Traditionally, systems were developed using a project-oriented approach. As a need arises, user-specific requirements are provided and a system satisfying them is developed. Large organizations, such as in the financial industry, have many such systems that are developed ad-hoc and integrated into huge existing infrastructures. Little or no documentation of these architectures exist, making it difficult to have an overview of the current systems and how they relate to one another.

Financial organizations cover a large domain, and are thus divided into various business units or sectors, each dealing with separate functionality. These business units all have separate processes, systems, architectures and infrastructures based on their own technology. Although many of the business units require the same or similar business functions and requirements, still each one has their own implementation for them.

For example, when a customer wants to apply for a savings account, he needs to fill in a form stating his personal details. These details are entered into a system, belonging to the savings accounts unit, and stored there. At another stage, when the same customer wants to apply for a credit card, he again has to fill in a form providing the credit card unit with his personal details. These details are entered and stored on another system, this time belonging to the credit card unit.

By using a project-oriented approach for developing software, systems are built for a specific purpose according to a single user's requirements. Thus, a single system is developed for a single user, as illustrated in Figure 1. Each business unit is responsible for developing and maintaining their own systems. As a result, the same requirements arise in more than one business unit, resulting in duplicate systems providing the same functionality, and being separately built in

each of the different business units. This duplication of systems consequently increases the development and maintenance costs of such large organizations considerably [14].

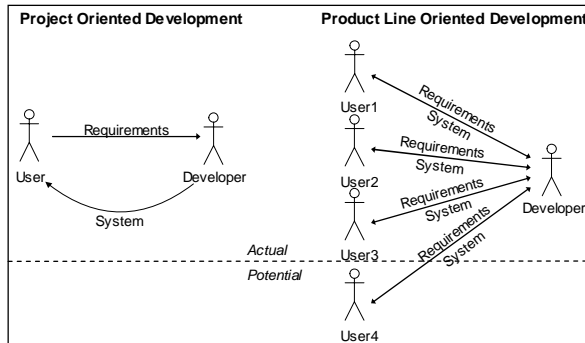


Figure 1. Project oriented vs. Product Line Oriented development

In order to reduce these costs, it is essential to introduce strategic reuse of existing assets into the organization. We propose a product line approach, where existing and potential future requirements are gathered from various users and are analysed. These requirements feed into the development of a product line, which is in itself a single product that can be configured to satisfy many users. As in a financial organization, various business units can use the same system to satisfy the required functionality. By using this approach, the development of duplicate systems is prevented and costs are considerably reduced.

1.2. Product Line Principles

Product line practice is used to facilitate strategic reuse in the development of software products targeted at a specific market. It consists of three main activities. They are:

- the development of core assets,
- the development of products, and
- the management of both these processes.

According to [4], “a software product lines is a set of software-intensive systems that share a common, managed set of features satisfying 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.” In order to develop highly reusable core assets, product line software engineering must have the ability to exploit commonality and manage variability among systems from a domain perspective.

Domain engineering plays an important role in the development of a product line. As stated in [7], it is “the activity of collecting, organizing, and storing past

experience in building systems or parts of systems in a particular domain in the form of reusable assets, as well as providing an adequate means for reusing these assets when building new systems.” Before reusable assets can be developed or used in a product line, the commonalities and variabilities of the products in the product line need to be identified by domain analysis. Once the common and variable features and their interdependencies are identified, they are represented in a feature model.

Feature modelling [10] is used in domain analysis to capture and organize the commonalities and variabilities [6] of systems in a domain in preparation for reuse access. The resulting feature model is used in designing the product line architecture [3], which must satisfy the general needs of the product line, as well as those of the individual products. As defined by [1], “the software architecture of a program or computing system is the structure or structures of the system, which comprise software components, the externally visible properties of those components, and the relationships among them.” The architecture, in turn is used to support product line implementation. After a product line is built and can evolve as the market need changes, it offers a great competitive advantage to the organization using it [4].

In the following section we state the problems that occur when developing systems in large organizations using a project-oriented approach. Section 3 describes the separation continuum and how it helps to identify and resolve some of the problem areas in large infrastructures. In Section 4 we propose the use of product line practices to facilitate strategic reuse in large financial organizations. Section 5 shortly describes the different dimensions that have been identified when implementing a product line in an organization.

2. The Problems of Project-Oriented Environments

Many organizations, especially in the financial industry, cover a large domain and therefore consist of separate business units responsible for performing different business functions. Each of these has their own infrastructure and systems running on different platforms. Most systems in these business units were developed using a project-oriented approach. As changes occur or new requirements arise, a system is developed to satisfy these requirements and is incorporated into the existing infrastructure of that business unit. Often, the same or very similar requirements arise across business units of an

organization. However, still each business unit develops its own systems to satisfy the requirements.

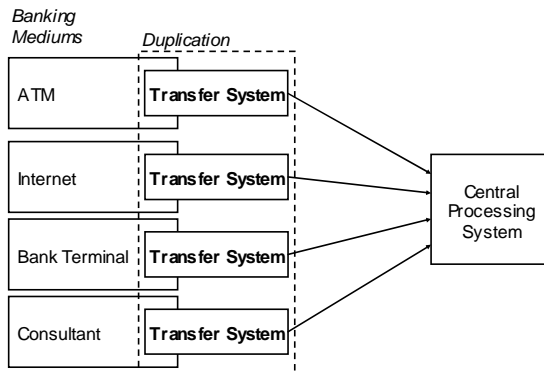


Figure 2. Money transfer example

In a banking environment, for example, a regular customer wanting to transfer money from one account to another, would have the option of doing such a transaction via various mediums, such as an ATM, the internet, a banking terminal or a consultant. Each medium has a separate infrastructure, and its systems are built using different technologies. The customer makes his transfer request via one of the mediums, which has its own specific *transfer system* that connects to a central processing system. Any one of the mediums can be used to process the same request, for example transfer money, yet they all have their own separate system responsible for handling the transfer request [14]. Figure 2 illustrates this concept.

Using a project-oriented approach for delivering products can yield several problems, especially in organizations aiming at being market-leaders in a specific domain. These organizations are structured internally into different levels [14]. For example, as illustrated in Figure 3, at the top is the Organization level followed by the Business unit, Project management and System levels.

Problems occur at each of these levels. At the Organization level, the organization's strategies and also the adoption of architecture strategies usually cause problems, as they are not properly defined. At the Business unit level, there are problems such as the time to market in introducing new products and product changes is delayed, costs of developing projects per business unit are high, and a separate infrastructure to support the business needs must be maintained. Resource contention and constraints, as well as minimal reuse and duplication of effort across projects are problems that occur at the Project Management level. At the system level, problems occur due to a high complexity associated with reusing systems across a

number of business units, as well as with the reusing of data and systems.

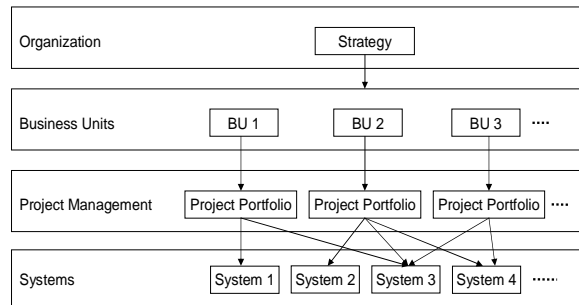


Figure 3. Project-oriented approach for delivering products

3. Dealing with Duplication by Separating Concerns

The separation of concerns has long been used by software engineers to manage the complexity of software development. By separating concerns, duplication can be easily identified and dealt with. The separation continuum, as defined by [13] is *“a systemic view of a system to understand its parts and their relationship, through understanding vertical and horizontal continuums needed where abstraction and implementation are on the vertical continuum, and human and machine facing aspects to have attributes of adaptability, context and relevance are on the horizontal continuum.”*

At the one end of the horizontal continuum are the visual aspects, whereas at the other end are the non-visual aspects of a system or family of systems. The horizontal continuum separates the user interface, the user interface controller, the connectivity, the business logic and also the data access from one another. These mostly only have an influence on the platform implementation aspects of the vertical continuum. At the one extreme of the vertical continuum are the abstract business concerns, whereas at the opposite extreme are the implementation details of the systems. The vertical continuum differentiates the layers needed to implement platform elements separately from the higher levels of abstraction needed at application levels, such as the business requirements. The levels along the vertical continuum are the business model, the system model, the applications portfolio, the services architecture, the component architecture and the object architecture.

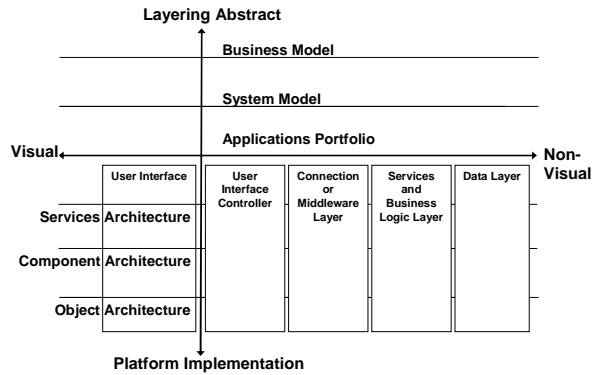


Figure 4. The separation continuum

Focusing more on the horizontal continuum, it becomes clear that each of the mediums mentioned above needs a separate user interface. An ATM, for example, has a very specific and limited user interface, while on the internet the graphical user interface representations are endless. It can also be deduced, that the user interface should be separated from the transfer system, which in turn should be separated from the central processing system. Figure 5 shows how the example of Figure 2 would be dealt with in this way. The figure also shows that a single transfer system can be used to provide the corresponding functionality for each banking medium.

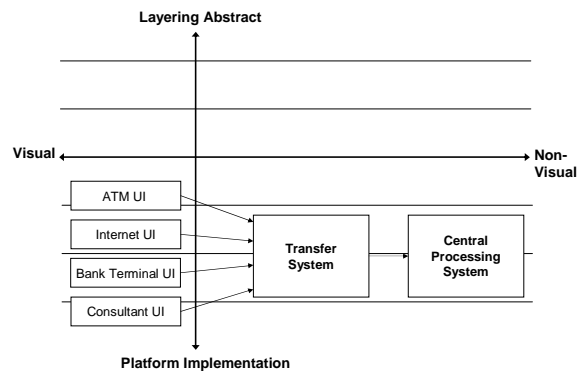


Figure 5. An example of separating concerns

There are great benefits to understanding interrelated layers of a system and the levels of re-usability required in producing software intensive systems [13]. One of the benefits is that highly adaptive software systems can easily respond to changes in the market environment. Another one is that the time-to-market is decreased, since new software products can effortlessly be assembled from existing reusable assets. Others are operational optimization and the ability to innovate.

4. Enabling Strategic Reuse

Product line practice can yield great benefits in large organizations, such as in the financial industry. Most of these organizations, however, have large existing infrastructures that were not built using a product line approach. Instead, they were built using a project-oriented approach, resulting in duplicated systems across numerous business units. Principles of product line practice need to be incorporated in order to facilitate strategic reuse of systems in these infrastructures.

Before a product line can be established in large financial organizations, the industry's domain needs to be analyzed. During domain analysis, information needs to be gathered about the functionality of every business unit in the organization, as well as the financial domain as a whole. Domain experts that have been involved in the financial industry for many years need to contribute to the domain understanding. However, throughout the domain analysis phase, it is most important to analyze the existing requirements received from a variety of sources in order to gain a thorough understanding of the financial domain. By analyzing the existing and potential future requirements, common and variable assets across the organization can be identified.

These commonalities and variabilities are represented in a high-level feature model, where a single feature is directly mapped onto an existing or potential asset. The feature model is essential for the design of the product line architecture, which supports the development of a single, organization-wide, common infrastructure. It distinguishes the common features, which are primarily existing assets needed to be integrated into the infrastructure, from the variable features, which are specific assets realized only through configuration management [5]. The common assets are strategically reused by including them in the product line architecture. They form the foundation of the infrastructure, which is the base of every product in the product line. These concepts are illustrated in Figure 6. Based on our literature survey these concepts have not been applied at a level of granularity that is most beneficial for large organizations. There is however, an example implementing the above mentioned concepts on a smaller scale, known as the MERGER product line as described in [4]. It started off from a single product known as Market Maker. Market Maker was a small system, which grew larger as its popularity

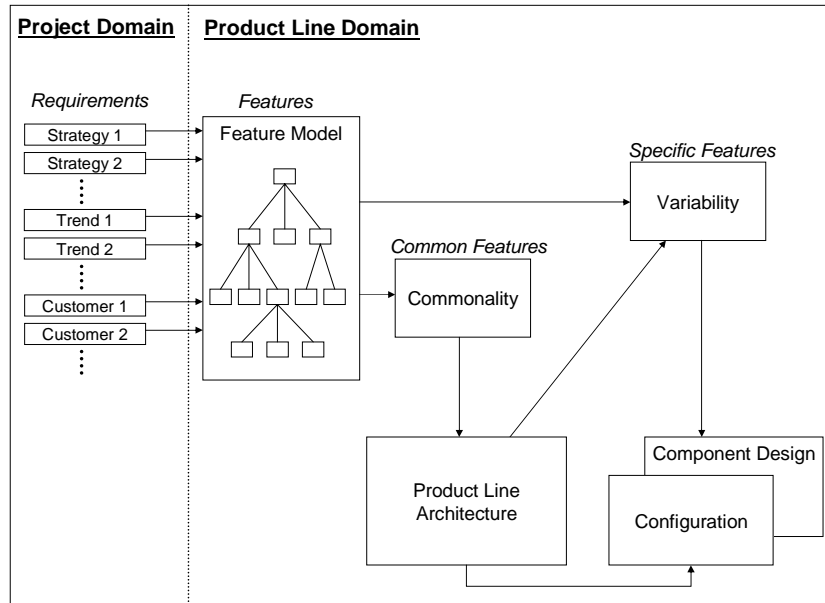


Figure 6. Implementing strategic reuse using product line practice principles

increased. With the growing demand, came the need to adapt the system to meet user specific requirements. In order to make the system easily customizable, it had to be flexible and adaptable. It was therefore decided to use a product line approach to build new products that have the same basic functionality but may have varying features. The latest version of the system, Market Maker 98, had to be reused in the MERGER product line. Rather than reusing its code, however, it was decided to reuse the extensive domain knowledge integrated in the existing product. Also, after implementation of the MERGER product line, customization of a product was not realized by changing any code, instead changes were realized solely by specific configurations of the variable assets.

5. Financial Services Product Lines

By developing a product line that introduces strategic reuse of existing assets, focus is shifted from building systems from disorganized user requirements in a project-oriented fashion to pro-actively planning of products that are to be released. This ensures that the majority of product releases are either delivering asset changes or applying changes to products by reconfiguring available assets. Reuse across product releases is maximized and duplication of effort across projects is eliminated. Time to respond to market opportunities is decreased due to the variation points that were anticipated in the product line. Also, product

delivery costs are reduced as products can be easily assembled from existing core assets, which are stored in and selected from a central repository.

There are three dimensions [14] that have been identified when implementing a product line in large organizations. These dimensions cover the product line, the core assets and the technology and systems, as illustrated in Figure 7. The existing systems, being software products related to a specific executable infrastructure, and the technology used to implement them represent the first dimension. The management of assets and products represents the two dimensions that follow.

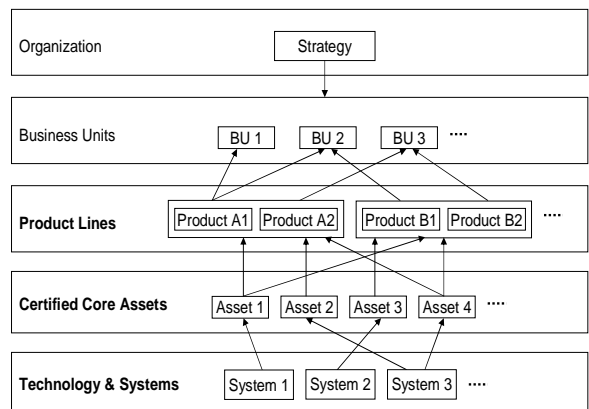


Figure 7. Using product lines for implementing strategic reuse across business units

The core assets form the basis of the production plan for products in the product line. These usually include a common architecture that all products in the product line will share, as well as software components that are developed for strategic reuse across the product line. Assets need to be certified [11] in order to provide a degree of confidence that ensures the success of the product line and its products. All products and assets are assessed on an ongoing basis to ensure their appropriate confidence. These are stored in a common repository, and are used to further develop the product line. The idea of certifiable assets is still a new one, and great interests lie in researching this concept.

The management of products concerns the implementation of products in the product line, by assembling the certified assets and through managing their configuration. These products can be used across different business units of the organization to provide the required functionality and satisfy the organizations goals.

6. Conclusions

Financial organizations have enormous software development and maintenance expenditures each year. The main reason for this is because of lack of analysis and planning. Being large organizations, they are run by individual business units that are responsible for providing different services. Each business unit functions with different systems and technologies in an independent infrastructure.

Systems are developed using a project-oriented approach, where a single system is developed to satisfy one user's requirements. A project-oriented approach allows fast, ad-hoc development of systems as the requirements arise. These systems are integrated into an infrastructure without much planning or any form of documentation, resulting in a close to chaotic structure that no person has an overview of. Huge amounts of money are spent trying to maintain such an infrastructure. As each business unit receives requirements independently to the next, it does occur that duplicate systems, providing the same functionality, are developed. It becomes clear that using a project-oriented approach drastically increases an organizations' software development and maintenance costs.

In order to reduce these costs, we have proposed using a software product line approach to facilitate strategic reuse of new and existing assets in large financial organizations. By analysing the requirements received across business units, commonalities and variabilities can be identified and represented in a

feature model. The features are mapped to high level assets, which need to be certified in order to provide a degree of confidence to the organization. Common assets are incorporated into the product line architecture, which forms the foundation of every product. Variability is only realized through the configuration of variable assets, allowing fast releases of products to be used across business units.

In future work, the effects of moving from an existing infrastructure to optimal reuse in an organization will be investigated. Before incorporating a product line-oriented approach into an organization's strategy, it is necessary to consider and evaluate certain trade-offs and issues that are involved in such an endeavour. Organizational restructuring is a large concern that needs to be carefully planned. Some surrounding issues are funding, staffing, integration and also technical issues.

7. Acknowledgements

Thanks to colleagues in the Polelo Laboratory for helpful discussions, especially John Müller and Cobus Smit. This work was funded by the NRF-DTI-THRIP grant no. 2788.

8. References

- [1] Bass, L., Clements, P., Kazman, R., *Software Architecture in Practice*, Addison-Wesley, December 30, 1997, ISBN 0-321-15495-9.
- [2] Bosch, J., "Maturity and Evolution in Software Product Lines: Approaches, Artefacts and Organization", *Software Product Lines: Second International Conference SPLC 2 Proceedings*, Springer, San Diego, CA, USA, August 19-22, 2002, pp. 257-271.
- [3] Bosch, J., "Product-Line Architectures in Industry: A Case Study", *Proceedings of the 21st International Conference on Software Engineering*, IEEE Computer Society, Los Angeles, CA, USA, May 16 - 22, 1999, pp. 544-554.
- [4] Clements, P., Northrop, L., *Software Product Lines – Practices and Patterns*, Addison-Wesley, August 20, 2001, ISBN 0-201-70332-7.
- [5] Conradi, R., Westfechtel, B., "Version models for software configuration management", *ACM Computing Surveys*, ACM Press, Vol. 30, No. 2, New York, NY, USA, 1998, pp. 232-282.
- [6] Coplien, J., Hoffman, D., Weiss, D., "Commonality and Variability in Software Engineering", *IEEE Software*, Vol.

15, No. 6, IEEE Computer Society, Nov/Dec 1998, pp. 37-45.

[7] Czarnecki, K., Eisenecker, U. W., *Generative Programming – Methods, Tools and Applications*, Addison-Wesley, June 6, 2000, ISBN 0-201-30977-7.

[8] DeBaud, J., Schmid, K. A., “Systematic Approach to Derive the Scope of Software Product Lines”, *Proceedings of the 21st International Conference on Software Engineering*, IEEE Computer Society, Los Angeles, CA, USA, May 16 - 22, 1999, pp. 34-43.

[9] Jacobson, I., Griss, M., Jonsson, P., *Software Reuse: Architecture, Process and Organization for Business Success*, Addison-Wesley, May 22, 1997, ISBN 0-201-92476-5.

[10] Kang, K. C., Lee, J., Lee, K., “Feature Oriented Product Line Software Engineering: Principles and Guidelines” Ch.2 of *Domain Oriented Systems Development: Perspectives and*

Practices, Taylor & Francis, UK, December 6, 2002, ISBN 0-415-30450-4.

[11] Smit, C., Müller, J., van Zyl, J., Bishop, J., “Towards Predictable Assembly of Certifiable Assets” Technical Report, University of Pretoria, Pretoria, South Africa.

[12] Speck, A., Clauß, M., Franczyk, B., “Concerns of Variability in "bottom-up" Product Lines”, Proceedings of Second Workshop on Aspect Oriented Software Development, Universität Bonn, Bonn, Feb 2002, pp. 19-24.

[13] van Zyl, J., “Product Line Architecture and the Separation of Concerns”, *Software Product Lines: Second International Conference SPLC 2 Proceedings*, Springer, San Diego, CA, USA, August 19-22, 2002, pp. 90-109.

[14] van Zyl, J., Continuous Research at SystemicLogic Research Institute, www.systemiclogic.com, Website accessed: May 2004.