Business Value through Product Line Engineering – A Case Study

[1]Devesh Sharma, [1]Aybüke Aurum, [2]Barbara Paech[1]

[1]*School of Information Systems, Technology and Management, University of New South Wales, NSW, Sydney 2052 Australia*
devesh.sharma@student.unsw.edu.au; *aybuke@unsw.edu.au*
[2]*Institut für Informatik, Neuenheimer Feld 326, 69120 Heidelberg, Germany*
paech@informatik.uni-heidelberg.de

Abstract

Software Product Line (PL) Engineering has been established in the last decade as a proven way to build flexibility and reusability into software systems. This approach is centred around the idea that the initial investments made in the development of reusable artefacts are outweighed by the quality and product improvements gained through the reuse of such artefacts. While there are many studies on introducing PL engineering into software development and calculating expected value upfront, there is little documented evidence of long-term experiences with PL engineering. This paper examines perceptions of the value of PL engineering for three PL of different ages. The paper confirms that, while PL engineering enhances product value and quality through shared components and architecture, this is also the predominant limitation of PL engineering. Furthermore, our investigations show that while perceptions of product quality differ depending on PL maturity, this is not the case for time-to-market and cost.

1. Introduction

The current trend of globalisation is placing increased pressure on software organisations to diversify their product offerings and maximise the business value of their software products. Over the last decade, software product lines (PL) [12] have emerged as a means for IT-related industries to address the diversity of the global market, and deliver their products in a timely and cost effective manner. The key idea behind PL is that most software organisations develop products in defined application domains, in which applications have more commonalities than differences. PL engineering (PLE) is a systematic development methodology that leverages this concept by building products from a common set of components and architecture

Value-based software engineering (VBSE) promotes approaches that maximise the business value (e.g. improving process, product or new business opportunities that increase profit or return on investment) for software organizations and provide a systematic process for analysing customer value [8]. Meeting customer needs and expectations is at the heart of developing any product or service. The ability to cater for multiple customers with different needs and requirements, such as through the use of PLE, is a powerful way of enhancing business value.

Over the last decade, numerous methodologies, such as Clement and Northrop's framework for SPL practice [12], FAST [22], Kobra [3], PULSE [6], and FORM [19], for PLE have been established. While all of these methods aim to standardise and improve PLE practices, few have specifically explored the business value that PLE entails. Some studies suggest that developing software as a PL brings competitive advantage and product improvements to the software organisations through lower production costs, rapid time to market, better quality, economic efficiency and better management of products [9, 13]; but such studies do not proceed to assess these claims.

This paper presents a comprehensive case study that aims to understand the benefits and limitations of PLE in software organisations and its ability to enhance business value. Results are presented from three PL at the Australian and Indian branches of a multinational software organization. The full details of the study

[1] The author participated in this work during her sabbatical at the School of Information Systems, Technology and Management at UNSW. The support given during that time is greatly acknowledged.

including further data on the effects of globalization on PLE can be found in [20].

The rest of the paper is organised as follows: Section 2 covers background knowledge on VBSE and PLE. Section 3 provides the details of the case study and Section 4 presents the design of the study. Section 5 presents the results of the data analysis. Section 6 provides a detailed discussion including validity threats and Section 7 concludes the paper.

2 Background and Related Work

2.1 Value Concepts in Software Engineering

Value creation is an economic activity. The development of value theory in economics has evolved from classical economics, such as work-value theory, to more modern theories, such as utility-value theory [22]. According to the utility-value theory, the value of a product can vary widely from customer to customer depending upon the importance they place on the various attributes of the product [7]. Utility-value theory also indicates that a customer's perceived value is different from the price of the product, and sometimes even independent of the price. Hence, a customer's perceived value represents the overall capacity of the product to satisfy a customer's needs, or the opportunity cost that the customer is willing to forgo in return for the product.

The value created in the production process is divided between customer and producer. The portion of the value received by the producer is the profit, calculated as the price of the product minus the cost of manufacturing the product. The portion of the value obtained by customer is the difference between the product value to the customer and the product's monetary price [7].

In software engineering, value creation involves gaining new insights or discovering new patterns of knowledge at the process level, product level or resource level [4, 8, 14]. The value of a product increases in proportion to its advantages over competitive products, and decreases in proportion to its disadvantages [2]. Thus the value of any product to a customer is a function of its performance and price, relative to other products on the market. The fact that software is different than other types of products only serves to complicate the matter [14]. Software is easily changed (in many cases, too easily) and is often rolled out several releases. Thus, it is not only a matter of looking at the short-term value of the next release; the long-term evolution of a software product has to be taken into account. There is a constant trade-off between short-term business goals to satisfy customers and different markets, and long-term evolution of the software to ensure that the software product is competitive in both the short and long-term.

For any given software product, there are many stakeholders involved in development, including product developers, management, sales and marketing staff, support engineers and the customers. Each stakeholder has a unique perspective of value [8]. For example, management in a software development organisation may measure value in terms of profit, whilst developers may find value in the robustness and quality of the software product. The customer may value the product in terms of its benefits and its cost.

It is the importance of incorporating these value considerations into software engineering practices that has lead to the development of PLE methods. PLE offers a systematic method for creating software product value through lowering development costs and through reuse of quality software artefacts. It also has the potential to improve a customer's perceived value of the product by constructing quality software products in a shorter timeframe.

2.2 Empirical Studies on SPL Business Value

Software product lines are as much about business practices as they are about technical practices. They require strategic thinking that looks beyond a single product [22]. PL have two distinct development processes, which are interlinked: domain engineering and application engineering [12]. Application engineering refers to the construction of software through systematic reuse of artefacts produced during the domain engineering stage. The application requirements engineers have to ensure both a high degree of reuse and the satisfaction of application stakeholders' needs. Thus, during application engineering it is especially important to balance value from the perspective of the developers and from the perspective of the customer and management.

There are several methods for determining the business value of PL upfront. Böckle et al., [10] present a method to calculate the return on investment (ROI) for PL. Faulk et al., [15] point out that PL can bring competitive advantage to the software company but it is important to calculate the risk for adopting PL and the risk inherent in applying PLE. Ganesan et al., [16] also include risk analysis and additionally consider PL generations. Wesselius [in 17] uses scenarios to address uncertainties about the future. Based on a case study, Baldassarre et al. [5] forecast the value of a PLE in terms of maintainability, extensibility and configurability from three different stakeholders' perspectives; namely customers, maintainers and producers. They confirm how PL contributes to stakeholders' value propositions. These methods are applied in case studies, but there is no evidence as to whether the forecasted business value really materialized.

Recently long-term experiences with PLE have also been reported. Deelstra et al., [13] analysed case studies on PL at a defence organization and a multinational electronics manufacturer, identifying problems for application engineering which diminish expected cost savings. Mansell [in 17] conducted a study of five small-and medium-sized organisations. They analysed established reuse practices and ROI in terms of investment and reduced development costs. The findings across the five companies varied, but overall the ROI was higher than the interest of a bank account. Kolb et al., [18] report insights from a company which show the value of PLE in spite of so far not achieved cost and time reductions. The value lies in the ability to offer a larger variety of products and in enhancing developer satisfaction. Ahmed et al., [1] empirically investigated the relationship between key organizational factors and PL performance in terms of cost and development time reductions, market growth, and financial strengths. The findings showed that organizational structure, culture, commitment, learning and change management are positively associated with the performance of PL. The paper does not provide data on performance, but analyses the influence of the organizational factors.

This research, taken together, shows that there is only emerging evidence of the overall business value of PLE. This study aims to add to this body of knowledge by investigating perceptions of PLE value in three PL of different ages.

3 Case Description

The object of this study was an international software development organisation operating in over one hundred countries around the world. It primarily specialises in database management systems, enterprise resource planning, customer relationship management and other industry-tailored products targeting government, finance and healthcare sectors. All of the company's products stem from its four major PL. The company has a few teams actively involved in development of its software in Australia, and a larger research and development base in India. Its headquarter is in the United States of America and employs over fifty thousand workers around the world, with more than twenty five percent of its workforce involved in software development. For the purposes of confidentiality, this organisation will be simply referred to as "the company" throughout this paper.

The company had four individual PL at the time this research was conducted. However, development in Australia was only conducted for three of the PL. Hence, only three PL could be examined in this study.

Product line A (PLA) comprises enterprise resource planning, supply chain management, customer relationship management, human resources and industry-specific applications targeting banking and healthcare. This is the company's original PL and has been undergoing iterative development for over ten years. It now boasts a large product mix, but has an aging core asset base with slow evolution of components and architecture. The products in PLA are currently in their twelfth major release.

Product line B (PLB) consists of a collection of products offering solutions for human resource management, customer relationship management, manufacturing, and student administration software for large corporations and government sectors. This PL was acquired by the company in 2005 through a takeover of its parent organisation. PLB has a relatively well maintained core asset base, and has a proprietary integrated development environment which forces developers to reuse core assets. The products in PLB are currently in the ninth major release, and its architecture is built around its own proprietary development platform.

Product line C (PLC) is the company's latest collection of products aimed at unifying the best-of-business capabilities offered by its applications and other PL. Through the use of an open, service oriented architecture, PLC is used as a standards-based technology blueprint that enables effective, predictable business process changes through standards-based integration of applications developed as web services. Developers and managers in PLC follow strict standards which do not allow for the duplication of core assets and encourage evolution of existing assets. To date, most PLC products are still undergoing development and have yet to be released.

4 Study Design

The primary objective of this study was to gain an understanding of how product value is created using PLE, and what factors enhance or reduce this value creation. This objective was further sub-divided into two separate research questions (RQ):

- RQ1: How is PLE currently used within the software organisation?
- RQ2: What are the benefits and limitations of PLE? How do they affect the value of software?

To take into account a customer's perceived value in addition to product value, the perception of the product quality was investigated. However, customers were not involved and the study was conducted purely from the perspective of the software organisation. Product value was investigated in terms of time-to-market and cost of production.

The study employed a mixed methodology; incorporating both qualitative and quantitative instruments to triangulate the results. The examination of each PL was conducted in two stages: semi-structured interviews and a short questionnaire.

The purpose of the interview was to qualitatively understand and describe the implications of PLE and the prevailing benefits and limitations. The interview examined 7 key areas, i.e. company and personal background details, product and PL background, reuse infrastructure and organisational practice, requirements engineering (RE) process, architecture design, product development and validation, product value creation and impact of global software development. Each interview took approximately an hour. The following interview questions focused on the issue of product value:

- How does PLE link to your organisation's business goals or strategies?
- Does PLE enhance your ability to deliver on customers' needs and requirements? If so, how?
- What impact has PLE had on your team's ability to develop product X?

An interpretive analysis of the interview data was conducted to address the research objective [21].

The short questionnaire was designed to quantitatively assess the perception of PLE delivering on their promised benefits of higher quality, reduced costs and reduced time-to-market. Participants were asked to assign a percentage out of 100 to each of the stated criteria.

Both research instruments were individually administered to the same participants in each PL.

The study involved 4 Australian participants from PLA, 4 Australian participants from PLB, and 2 Indian and 1 Australian participant from PLC. The participants were product and development managers.

5 Results

In this section we discuss the results of the different PL individually. First we concentrate on the overall setting (RQ1) and then on perceptions of value (RQ2).

5.1. Description of the Situation of PLA

PLA has a large number of repositories in which reusable software assets are restored. The assets range from requirements documents through to source code and test cases. All participants were aware of the repositories available for the storage of software artefacts of their individual products; however some participants questioned their effectiveness. PLA shares these repositories across the entire PL. All artefacts common across all the products are kept in a single repository, and the artefacts used by specific products are kept in separate repositories.

PLA has established tools, which handle the configuration management of such assets. All software artefacts have to undergo an established change management procedure requiring multiple levels of review. These procedures are built into a formal development methodology used across PLA. However, PLA does not have established procedures to identify reusable artefacts and does not have formal decision-making criteria to assess an artefact's suitability to be placed into a reusable asset to repository. PLA does not have any formal methods for calculating ROI from its reusable software artefacts and development procedures.

The RE process used was typical of one used for single system development with no formal mechanism for advancing any requirement to become PL requirements. For all products in PLA, requirements are elicited from stakeholders of the developed application and then compiled into a requirements definition document after negotiations between functional teams and a central strategy team. Selection and prioritisation of requirements in PLA is heavily influenced by its core customers, and importance is placed on customers that generate large amounts of revenue. However all four participants acknowledged that it was also important for products to be "best of breed" and market leaders. Time, cost and resources were treated as secondary criteria by most participants. Two participants indicated that maintaining market dominance had a greater impact on requirement prioritisation. Existing reusable software artefacts were generally ignored during the selection and prioritisation of requirements.

All applications in PLA are developed from a common architecture. While all applications must conform to this architecture, lead developers have some autonomy over the design of individual products by building on top of the PL architecture. Using the common architecture enforces standards on the applications that are developed from it, and two participants felt that this places technical constraints on what they can develop and the functionality that can be delivered to the customer. Most of the applications in PLA have been developed over a number of years, and newer releases are put through an iteration phase. The product architecture from previous releases is reused for later releases, and components selected are modified through mechanisms of inheritance or parameterisation. Selection of appropriate components is generally left to the knowledge and experience of developers. All developed applications also undergo multiple stages of testing to verify their adherence to the original requirements, progressing from unit and integration testing conducted by development teams, through to testing by a quality assurance team.

5.2. Description of the Situation of PLB

PLB has a systematic method of storing reusable software assets. Common objects shared across the PL are stored in a central database and a proprietary integrated development environment is used to call these objects. Product specific classes are also stored in separate product repositories. All documentation is stored in an online document management system. Configuration management of artefacts in PLB is managed through an established change management cycle and controlled through the use of proper configuration management tools. The change management cycle was standardised across PLB through its development methodology.

PLB also had ad hoc procedures and decision-making criteria to assess an artefact's suitability to be placed in the reusable asset repository. The flexibility and generic nature of a component was of critical importance in deciding whether it could be reused, and to be placed in the central database. None of the participants could identify any formal methods for calculation of ROI for the reusable infrastructure. One participant questioned whether it was possible to calculate ROI for individual components.

All products in PLB followed the standard RE process. The process is similar to that used for PLA, with requirements elicited from stakeholders being assessed for their feasibility by a central strategy team. The requirements are be placed into a requirements definition document, which is forwarded to product and development managers. Product managers are given the responsibility of producing a functional design document, while developing managers are responsible for producing a technical design document. The requirements for products in PLB are both market and customer driven. Resource constraints, time, complexity of development, and customer demand were cited as the major criteria in the selection and prioritisation of requirements. Higher management and the product strategy teams were responsible for requirement selection and prioritisation, which were subsequently reviewed by a customer focus group. All participants viewed cost as a secondary criteria when selecting and prioritisation requirements. Reusable software artefacts were given active consideration in the requirements selection and prioritisation process. One participant added that some requirements could be considered for the entire PL based on their generic nature and their ability to be modified for individual product. In PLB, requirements that cannot be catered for by the existing architecture are considered when a product is made more marketable and competitive. One participant stated that architecture enhancement requests can be made for such requirements. Any requirements which could not be satisfied altogether are delayed for future releases.

Every product in PLB is developed from a common architecture with common functionality being stored in a central database. As all applications are developed in a proprietary integrated development environment, they are forced to use this common functionality. The PL architecture is published in technology manuals and white papers. The restrictive nature of the PL architecture was a result of it being shared across multiple products. Two participants indicated that sharing the architecture across many products makes it difficult to change and maintain. Most development teams reuse their source code to develop later releases. Components are seldom modified and are generally passed parameters to alter their behaviour. In a similar manner to PLA, importance is placed on the functional and technical design documents, and certification testing ensures that the product is compatible with the PL.

5.3. Description of the Situation of PLC

All applications in PLC have been built using Service Oriented Architecture (SOA) techniques, whereby applications are built out of software services. These 'services' are large units of functionality, such as placing an order for goods, and have no association to any other services embedded in them. One participant indicated that the primary objective in PLC was to have a single instance of every component or service to avoid replication and ensure reuse. Software artefacts created for PLC products are categorized and stored, and are shared across the PL. Using SOA, PLC aims to eliminate product specific components as all service objects can communicate to each other through the use of metadata and standard communication protocols. PLC utilises automated configuration management tools which are merged with the development environments to manage changes in software assets. It also uses a defined change management procedure. Unlike PLA and PLB, service objects created in PLC may not necessarily be product specific. Because all objects in PLC are developed as services, development of software becomes a matter of utilising the right services to process data as required, and services may be easily used across multiple products. None of the participants could describe any methods they use to calculate ROI for reusable assets. Two participants were unaware of any such process, but the third participant outlined that an analysis was conducted for all objects before they were developed.

Although most products in PLC are still undergoing development, an established RE process was being followed by development and product managers. The method is very similar to PLA and PLB, with a central strategy team gathering requirements, subsequently publishing feasible requirements in a requirements definition document. Product and development managers are then responsible for creating functional and technical designs respectively. One participant explained that requirements came from a large number of sources: The selection and prioritisation of requirements in PLC is influenced by internal priorities, resource constraints and a mix of customer and market demands. Only one participant cited development cost as a criterion in prioritisation of requirements. Since PLC is in its development stage, a process for managing requirements beyond the existing architecture could not be defined.

The architecture used in PLC is extremely flexible as SOA services are loosely coupled. Unlike products in PLA and PLB, applications in PLC are not compiled into an executable with rigid links. Senior architects are responsible for developing links between service objects to provide the required functionality and solutions. PLC used a layered approach in its architecture with any product-specific objects communicating to other generic services through metadata. None of the participants could identify any functional or non-functional constraints that were placed on their product by architecture. Applications in PLC are still undergoing initial development, which involves selecting the right service objects and provisioning for interaction between the service objects. There is limited modification of existing components as the functionality of applications could be simply modified by selecting different instances of components. All applications developed in PLC were tested in a similar manner to PLA and PLB.

5.4 Product Value Perception in PLA

All participants in PLA understood that the primary objective of the company was to generate revenue from its products. PLE helped to improve revenue streams from the efficient development of software. PLE was also identified as enhancing the ability to deliver on customers' needs and requirements. One participant stated that PLE allowed for greater integration between products and helped to better deliver on customer needs.

All participants perceived PLE as having a positive impact on their ability to develop products. PLE was perceived as a means of enabling standards based development that allowed for the integration of products. However, this also induced limitations. One participant stated, referring to the SPL architecture: *“If there is something new out there we cannot use it because of the restrictions we face, because the new functionality hasn't been certified, it hasn't been tested to the product line architecture and its principles. It doesn't always do us any good, and creating unnecessary constraints and creating a lot of extra work”*.

5.5 Product Value Perception in PLB

In PLB, all participants understood that the ultimate goal of the company was to deliver on its customers' needs and requirements. PLE had enabled the company to be focused on their customer's needs and requirements. PLE was also directly associated with the development of products that satisfied customer requirements and delivered products in a shorter time frame. One participant stated that PLE enabled his team to focus on the development of new functionality. Another participant also indicated that PLE enabled his team to reduce development time due to its ability to focus development on a focused scope,

All participants had a general perception that PLE had a major impact on their product development and increased the their team's efficiency in developing products.

The primary limitation of PLE was seen in its restrictive nature of shared components and architecture. Sharing architecture across many products made it difficult to change and maintain, and limited the functionality of products.

5.6 Product Value Perception in PLC

Most participants in PLC described the company's business goals as a combination of satisfying customer needs and generating revenue from its products. One participant indicated that both objectives were essentially interrelated as customer satisfaction lead to market domination, which was the primary determinant of revenue generation. PLE was identified as a means of enhancing the ability to deliver on customers' needs and requirements. PLE helped the team deliver quality products to customers:. All participants felt that PLE had a positive impact on the team's ability to develop their respective products.

PLE ensured the development of quality products through the use of common standards.

The grouping of products inevitably leads to large scale interdependencies, but despite having common standards, a single failure can lead to large scale delays in development. One participant outlined how this can occur in PLE: *"(The) negative aspect is that when we use low level components, and there are issues and bugs in that, we need to change our timelines to incorporate the time taken to fix that component"*.

5.7 Results from the Questionnaire

As can be seen from the last three sections, despite of the differences in the settings, the participants of all SPL confirmed that PLE links to the organisation's business goals, enhances the ability to deliver on customers' needs and requirements, and supports the team's ability for development. After the conclusion of the interview, each participant was asked to rate the ability of PLE to reduce time-to-market, reduce costs and increase quality as a percentage out of 100. The answers for each criterion from each PLE were summed and an average obtained. The averages obtained represent the estimated ability of PLE to deliver benefits to the software organisation compared to the standalone development of the individual products contained within PL. The results are presented in Table 1.

Table 1. Perceived Benefits of PLE

Product Line	Reduced Time-To-Market	Higher Quality	Reduced Development Costs
A	70%	50%	70%
B	72%	70%	67%
C	73%	67%	63%

The main difference is that participants in PLA did not perceive PLE as a strong means of producing higher quality products. Based on the study we cannot establish the reasons for this difference. However, as described in section 5.1-5.3, the main difference among the three PL was the age and the enforcement of the reuse standards. PLA shows an aging core asset base and slow evolution of core assets, which in turn limits the functionality delivered from newer releases of products in the PL. Comparatively, PLB and PLC apply a much more forced reuse and continued evolution of core assets. Thus, it may be inferred that the quality delivered by PLE is dependent upon the quality of the core assets themselves.

6 Discussion of Results

This exploratory study brings important empirical knowledge and evidence to the PLE domain and provides diversified views on product improvements from a rich case study. The following three key observations are made from this study:

Firstly, PLE was generally perceived as having a positive impact on product development and business value across the three PL. While all participants recognized that the primary business objective of the company was revenue generation, there was some difference of opinion as to how PLE helped achieve this objective. In PLA, the ability of PLE to focus development on a defined scope and deliver on customer requirements was seen as conforming to the company's business objectives. This view was also expressed in PLB and PLC, which went on to further describe cost savings delivered by PLE through the "efficient" use of development staff.

Secondly, the ability of PLE to integrate multiple products was seen as both an advantage and disadvantage. It was recognized in PLA that the integration of products could satisfy broader customer requirements that go beyond the scope of individual applications. The ability to integrate products and share common components was also perceived as reducing time and increasing efficiency in PLB. However, the reuse of common components and the strong integration of products is a double edged sword. It was indicated in PLC that any faults in these common components had the potential to cause delays across the entire PL. This was reinforced in PLB and PLA.

Thirdly, PLE was also perceived as a means of encouraging focused development due to the defined scope and domain used in a PL.

Overall, it can be concluded for all three PL that while PLE enhances product value and quality through shared components and architecture, this is also the predominant limitation of PLE. Furthermore, it shows that while perceptions of product quality differed depending on PL maturity, this was not the case for time-to-market and cost.

The results of this study must be interpreted with caution as the three PL studied are not a representative sample of the software industry. The participants selected may not adequately reflect the diversity of opinion on the current practice of PLE. This research has low external validity as it only looks at three PL from a single software organisation. The small sample size used in this study also indicates that conclusions made may not be. However, the company is very typical for a large software development company, and has sustained experience with PLE. A potential threat to the internal validity of this research is related that is possible that the respondents of the questionnaire may not have understood the questions as intended, or in a similar manner to one another.

7 Conclusion and Future Work

This paper presents an exploratory study in providing a more comprehensive picture of business value of PLE. It provides the first steps towards a greater understanding of the product improvements delivered through systematic reuse and processes within the software organisation.

We have tried to add empirical evidence on the long-term effects of PLE, particularly with respect to business value. Similar to previous studies, for all three PL a positive impact on business value was observed in terms of meeting business goals and customer needs, in addition to reduced time-to-market, reduced cost of product development and improved team development. In other words, PLE seemed to enhance product value and quality through shared components and architecture; however, this was also the predominant limitation of PLE. Furthermore, the results showed that while perceptions of product quality differed depending on PL maturity, this was not the case for time-to-market and cost.

Future work should focus on more thoroughly investigating the different facets of business value. Empirical studies to date have only looked at the different facets individually and it is important to link them together. In particular, perceptions of product quality requires further investigation.

References

1. Ahmed F, Capretz LF, Sheikh SA (2007) Institutionalization of Software Product Line: An Empirical Investigation of Key Organizational Factors. Systems and Software 80(6) :836-849
2. Alwis, D., Hlupic, V., Fitzgerald G (2003) Intellectual capital factors that impact of value creation. 25th Int. Conf Information Technology Interfaces, Croatia, pp 411-416
3. Atkinson, C. et al., (2002) Component-Based Product Line Engineering with the UML Addison-Wesley, New York
4. Aurum A, Wohlin C (2007) A Value-Based Approach in Requirements Engineering: Explaining Some of the Fundamental Concepts. Int. Conf on Requirements Engineering: Foundation for Software Quality (REFSQ'07), 11-12 Trondheim Norway.
5. Baldassarre MT, Caivano D, Visaggio G (2006) Software Product Lines in Value Based Software Engineering. 10th Int Conf on Evaluation and Assessment in Software Engineering (EASE) Keele Univ, UK, 10 - 11 April 2006
6. Bayer J, Flege O, Knauber P, Laqua R, Muthig D, Schmid K, Widen T (1999) PuLSE: A Methodology to Develop Software Product Lines. Proc. of the 5th ACM SIGSOFT Symposium on Software Reusability (SSR'99), pp. 122-131
7. Besanko D, Dranove D, Shanley M, (2000) Economics of strategy, John Wiley & Sons, New York, 2nd Edition
8. Biffl S, Aurum A, Boehm B, Erdogmus H, Grünbacher P (Eds.) (2005) Value Based Software Engineering, Springer, Germany
9. Birk A, Heller G, John I, Schmid K, von der Masen T, Muller K (2003) Product Line Engineering: The State of the Practice. IEEE Software Nov/Dec
10. Böckle G, Clements P, McGregor JD, Muthig D, Schmid K (2004) Calculating ROI for Software Product Lines, IEEE Software May/June, pp. 23-31
11. Browning TR (2003) On Customer Value and Improvement in Product Development Processes. Systems Engineering, 6(1):49-61
12. Clements P, Northrop L (2002) Software Product Lines : Practices And Patterns, Addison-Wesley, Boston
13. Deelstra S, Sinnema M, Bosch J (2004) Experiences in Software Product Families: Problems and Issues During Product Derivation. Proc. of the 3rd Software Product Line Conference, pp. 161 - 182
14. Favaro J, Favaro KR, Favaro PF (1998) Value Based Software Reuse Investment. Annals of Software Engineering, 5: 5–52
15. Faulk SR, Harmon RR, Raffo DM (2000) Value Based Software Engineering: A Value-Driven Approach to Product Line Engineering. 1st Int. Conference on Software Product Line Engineering, Denver, Colorado, USA
16. Ganesan D, Muthig D, Yoshimura K (2006) Predicting Return on Investment for Product Line Generations. 10th Int. Conf on Software Product Line Engineering, pp.13-24
17. Käkölä T, Dueñas JC (Eds.) (2006) Software Product Lines: Research Issues in Engineering and Management, Springer, Germany
18. Kolb R, John I, Knodel J, Muthig D, Haury U, Meyer G (2006) Experiences with Product Line Development of Embedded Systems at Test AG, .10th Int. Conference on Software Product Line Engineering
19. Lee J, Kang KC, Kim S, Kim K, Shin E, Huh M (1998): FORM: A Feature-Oriented Reuse Method with Domain-Specific Reference Architectures, Annals of Software Engineering, 5:143-168
20. Sharma D (2007) Blueprint of Success: Creating Software Product Value through Product Line Engineering. Honours Thesis. School of Information Systems, Technology and Management, University of New South Wales
21. Silverman D (2001) Interpreting Qualitative Data. Sage Publication, London UK
22. Weiss DM, Lai C (1999) Software Product-Line Engineering: A Family-Based Software Development Process, Addison-Wesley
23. Yannou B, Ahmed WB (2003), Polysemy of Values or Conflict of Interests: A Multi-Disciplinary Analysis. Int. Journal of Value-Based Management, 16(2): 153-179