# Comparative Study Between Software Product Line and Waterfall Process

Waraporn Jirapanthong

Faculty of Information Technology, Dhurakij Pundit University, 110/1-4 Prachachuen Road, Laksi, Bangkok 10210, Thailand, waraporn@it.dpu.ac.th ,+66(0)9547300 ext. 385

**ABSTRACT –** Software product line has been recognised as an important paradigm for software systems engineering. In the last years, a large number of methodologies and approaches have been proposed to support the development of software systems based on product line development. However, its context leads difficulties to software product line engineering in practical. It has been quested whether software product line-based approach is more productive and flexible than traditional software development model i.e. waterfall model. This research thus examines the qualitative and quantitative aspects of software development which applies software product line and waterfall. The paper presents the study on empirical projects based on software product line and waterfall processes. In particular, we conducted the survey and interview to capture the satisfaction of stakeholders and measured the effort spent during software development and maintenance.

**KEY WORDS –** Software Product Line, Product Family, Waterfall, and Software Measurement

## 1. Introduction

Nowadays, many software development projects focus on customer satisfaction, quick adaptation to changes, and flexibility. Therefore, software product line development has become popular because it responds well to frequent changes in user requirements. Software product line shares a common set of features and are developed based on the reuse of core assets have been recognised as an important paradigm for software systems engineering. Recently, a large number of software systems are being developed and deployed in this way in order to reduce cost, effort, and time during system development. Various methodologies and approaches have been proposed to support the development of software systems based on software product line development.

Although software product line development is criticized as having difficulties, it has been more popular. Some difficulties are concerned with the (a) necessity of having a basic understanding of the variability consequences during the different development phases of software products, (b) necessity of establishing relationships between product members and product line artefacts, and relationships between product members artefacts, (c) poor support for capturing, designing, and representing requirements at the level of product line and the level of specific product members, (d) poor support for handling complex relations among product members, and (e) poor support for maintaining information about the development process.

This research, thus, examines the qualitative and quantitative aspects of software development using software product line architecture, in comparison with those using a traditional software model, *waterfall model*. In particular, the study used both qualitative aspect that were collected from surveys and interviews of development and maintenance team and quantitative aspect that were measured from effort spent during the development and maintenance phases.

## 2. Background

This section presents background material on software product line, waterfall model, and software metrics.

### 2.1 Software Product Line

Software product line systems share a common set of features and are developed based on the reuse of core assets. A family of software systems are developed and deployed in this way in order to reduce cost, effort, and time during system development. Various methodologies and approaches have been proposed to support the development of software systems based on product line development. Examples of these methodologies and approaches are FeatuRSEB [6], FAST [14], FORM [10], FODA [9], PuLSE [2], and KobrA [1].

The above methodologies and approaches are also known as *domain engineering* approaches and emphasise a group of related applications in a domain, instead of single applications. Their main focus is the identification and analysis of commonality and variability principles among applications in a domain in order to engineer reusable and adaptable components and, therefore, support product line development.

There are three steps for domain engineering: (a) *domain analysis* is the process of identifying, collecting, organizing and representing the relevant information in a domain, based upon the study of existing systems and their developing histories, knowledge captured from domain experts, underlying theory, and emerging technology within a domain [9]. Software artefacts that are produced during the activity of domain analysis are called *reference requirements*, which define the products and their requirements in a family. The reference requirements contain commonality and variability of the product family. The activities occur during the domain analysis are scoping, defining of commonality and variability, and planning for product members and features.

(b) *domain design* is the process of developing a design model from the products of domain analysis and the knowledge gained from the study of software requirements or design reuse and generic architectures. Software artefacts that are produced during the activity of domain design are called *software product line architecture*, which forms the backbone of integrating software systems and consists of a set of decisions and interfaces which connect software components together. Software product line architecture differs from an architecture of single systems that it must represent the common design for all product members and variable design for specific product members [11]. The activities occur during the domain design are defining and evaluation of software product line architecture.

(c) *domain implementation* is the process of identifying reusable components based on the domain model and generic architecture [4]. Software artefacts that are produced during the activity of domain implementation are called *reusable software components*. The activity is focused on the creation of reusable software components e.g. source codes and linking libraries that are later assembled for product members At the end of the domain engineering process, an organization is ready for developing product members.

Additionally, *application engineering* is a systematic process for the creation of a product member from the core assets created during the domain engineering. Domain engineering assures that the activities of analysis, design and implementation of a product family are thoroughly performed for all product members, while application engineering assures the reuse of the core assets of the product family for the creation of product members. There are activities such as: (i) *requirements engineering***,** which is a process that consists of requirements elicitation, analysis, specification, verification, and management; (ii) *design analysis***,** which is a process that is concerned with how the system functionality is to be provided by the different components of the system*;* and (iii) *integration and testing*, which is a process of taking reusable components then putting them together to build a complete system, and of testing if the system is working appropriately.

However, although the support for identifying and analysing common and variable aspects among applications and the engineering of reusable and adaptable components are important for software product line development, they are not easy tasks. This is mainly due to the large number and heterogeneity of documents generated during the development of product line systems.

## 2.2 Waterfall Model

The Waterfall model is originally invented by Winston W. Royce in 1970. This model assumes that requirements remain static throughout a project and emphasizes having documents to support each development step. Waterfall development has distinct goals for each phase. The process has five steps: (a) requirement definition, (b) software and system design, (c) implementation, (d) integration and testing, and (e) operation and maintenance. Software industry in Thailand is generally based on waterfall model which emphasis on documentation such as user requirement specification, testing document and design diagram. The developers spend time to create and update documents in order to cover entire of a software project. Therefore, many documents are created during software development process. In addition, changing of requirement is not flexible. However, the documents are expected to be used by developers and stakeholders. Those also support new stakeholders who later join the software project.

## 2.3 Software Metrics

Measurement is the process by which numbers or symbols are assigned to attributes of entities in the real world in such a way as to describe them according to clearly defined unambiguous rules [13]. Entities are such as software design, software design specification, software code, and software development team. Attributes are such as defects discovered in design review, number of pages, number of line of code, number of operations, and team size, average team experience.

Types of metric are several. The common type of metric is product metric. The generally accepted measures for the software product are size and quality. In particular the size measure of software product involves two ratios: (a) line of code (LOC) and (b) function point (FP) [13]. The quality measure of software product involves maintainability measurement such as coding effort, design effort, percentage of modules changes, classes changes, classes added. In addition, [5] presents the maintainability metrics in external view such as mean or median time to repair, ratio of total change implement time to total number of changes implemented, number of unsolved problems, time spent on unsolved problems ,percentage of changes that introduce new faults and number of modules modified to implement a change.

## 3. Research Method

The goal of this research is to compare the qualitative and quantitative aspects between software product line -based and waterfall-based development and maintenance. To achieve the goal, this research conducted an experiment involving three software development projects that have similar requirements and some different requirements. A team of developers was required to achieve the software development projects two times: (i) to follow the software product line process, and (ii) to follow the waterfall process.

### 3.1 Empirical Project Development based on Software Product Line Process

This project started with developers training in software product line processes and techniques. These developers were then tested for their understanding of software product line practices by using questionnaires. Those who passed the test were assumed to be ready to implement projects using software product line. The developers then started developing a set of three projects by following the software product line practices. They studied and analyzed all projects together and produced the following software artefacts:

- reference requirements
- software product line architecture
- software components

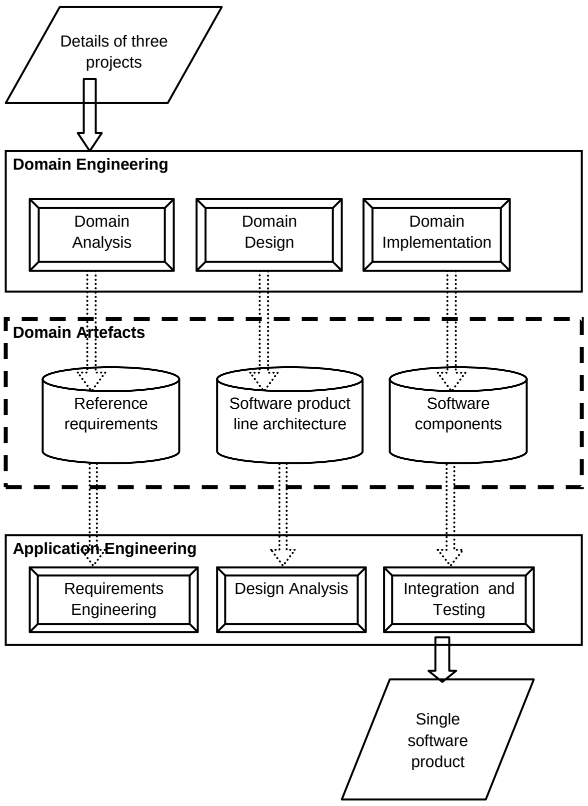

**Figure 1:** Software Produce Line Process

The artefacts were checked before submitting to the domain repository to be ready for application engineering process. Next, three software products were created based on the domain artefacts (i.e. reference requirements, software product line architecture, and software components). Before the software was accepted by customers, we ran test cases on the software. When the software passed all test cases, the projects are completed. The whole software product line process is shown in Figure 1.

We then calculated and analyzed the qualitative and quantitative aspects of domain engineering process and application engineering process for each project. Then we checked the developers conform to software product line practices.

### 3.2 Empirical Project Development based on Waterfall Process

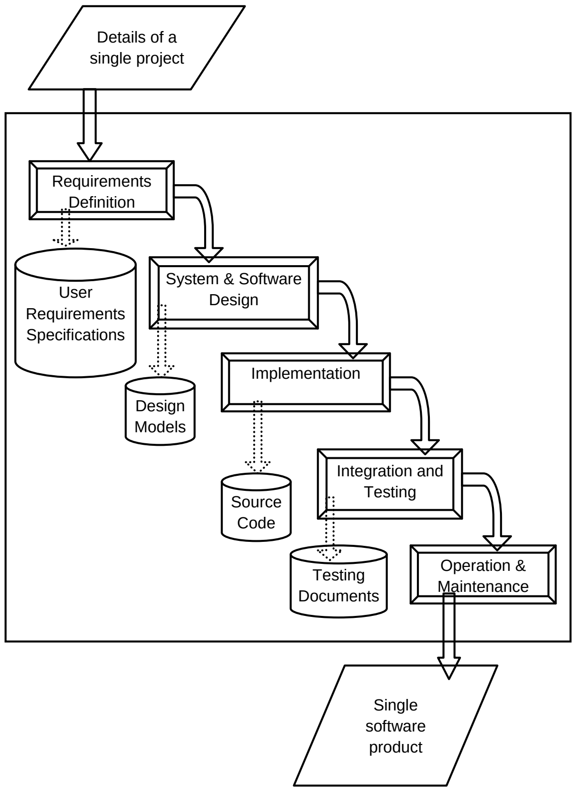

**Figure 2:** Waterfall Process

For each project, developers divided their work based on their roles. Firstly, the developers summarized all requirements from customers and produced a user requirement specification. Next, they designed the system architecture, components and data models. They applied use case descriptions and diagrams to explain the requirements of each single software product. In addition, they also created class diagrams, sequence diagrams and activity diagrams of the entire project in this stage. They implemented the software by following the documents and used unit tests regularly. When completing all the components, the developers integrated all the pieces together and began an integration test. Finally, the developers delivered the customers the complete software when all of these stages finished. Figure 2 shows the flow of these projects. The artefacts that are checked and submitted to the repository are:

- use case descriptions
- use case diagrams
- class diagrams
- sequence diagrams
- activity diagrams
- source code
- testing documents
- coding standard and technical documents

The project that used waterfall-based model produced more artefacts than that of software product line process does. However, the development time of the waterfall-based project is greater than that of software product line. All the end of this step, we calculated and analyzed the qualitative and quantitative aspects in each project.

## 3.3 New Requirements Management on Software Products

In this phase, the team of developers was given new requirements on the systems. Many factors lead into this scenario, for example, customers require new functionality to be done in a design part of a software product.

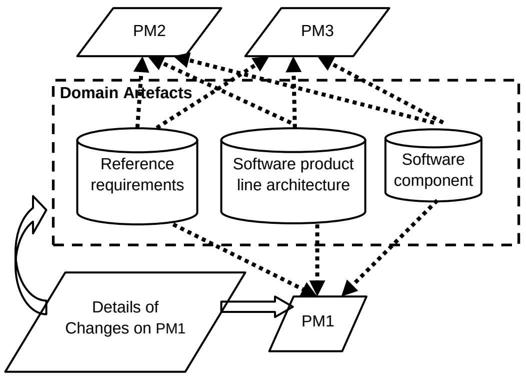

**Figure 3:** Empirical Project Maintenance

For the software product line-based systems, it is supposed the situation in which the organisation has established a software product line for their software systems with software product members. Those are created from the development phase. And the new requirements are done to a product member. Therefore, it is necessary to evaluate how these new requirements will affect the other artefacts of the product member and if these new requirements also affect other product members in the software product line that may be related to the new requirements. The artefacts are inspected and determined if they are related to the new requirements as shown in Figure 3.

For the waterfall-based systems, it is supposed the situation in which the organisation has individually developed a set of software systems. Those are created from the development phase. And the new requirements are done to a software product. Therefore, it is necessary to evaluate how these new requirements will affect any artefacts of the software product.

## 4. Experiments

According to the experiments, we captured the time spent during development of three software products by following those software process models. Those software products are having similar requirements and some differences. The developer team was required to develop a set of three software products two times. Each time each different software process model was followed.

### 4.1 Project Characteristics

The developers were requested to develop three software products, namely PM1, PM2, and PM3. The list of functionalities and specifications of each software product is shown in Table 1. We describe below the details of each product.

**PM1**

The product member PM1 is expected to be a software system which supports an accounting department and is used by young people. As shown in Table 1, the product member PM1 has some basic functionalities: (a) *processing accounting data*, (b) *generating a report*, (c) *managing log files*, and (d) *displaying time and date*. Additionally, it has some advanced functionalities: (a) *access Internet* which allows a user to browse and download data through the Internet, and (b) *email system* which supports the email.

**PM2**

The product member PM2 supports work related to accounting and offers a simple design and is targeted for users who are not familiar using a computer. The product member PM2 has only basic functionalities: (a) *processing accounting data*, (b) *generating a report*, (c) *managing log files*, and (d) *displaying time and date*.

**PM3**

The target customers of the product member PM3 are users who work via the Internet. The system supports web-based environment. As shown in Table 1, the product member PM3 has some basic functionalities: (a) *processing accounting data*, (b) *generating a report*, (c) *managing log files*, and (d) *displaying time and date*. In addition, PM3 offers advanced functionalities: (a) *access Internet* which allows a user to browse and download data, (b) *email system* which supports the email, (c) *send and receive text messages*, and (d) *send and receive multimedia messages.*

**Table 1 shows the functionalities of each software product**

| Functionality | PM1 | PM2 | PM3 |
|---|---|---|---|
| F1 | X | X | X |
| F2 | X | X | X |
| F3 | X | X | X |
| F4 | X | X | X |
| F5 | X |  | X |
| F6 | X |  | X |
| F7 | X |  | X |
| F8 |  |  | X |

F1: Processing accounting data
F2: Generating a report
F3: Managing log files
F4: Displaying time and date
F5: Enabling access the Internet
F6: Enabling emailing
F7: Sending and receiving text messages
F8: Sending and receiving multimedia message

### 4.2 Software Product Line -based Projects

*A. Development Phase*

The projects have been developed based on study, analysis, and discussions of business domain. Software systems are created based on demands which require a variety of software products. In this way, a number of documents are created by developers. The team of developers analysed and designed a family of software systems with three members. Each member has shared and specialized functionalities with the family. The product members are aimed to satisfy different targets of customers.

In addition, reference requirements is produced and documented in term of a feature model as software product line architecture is produced and documented in terms of subsystem, feature, and process models [7]. The following artefacts are created:

(a) a feature model is created and composed of common features representing mandatory features, alternative and optional, representing different features between product members. For example, all product members must provide features of processing accounting data, generating a report, managing log files, and displaying time and date.
(b) a subsystem models is created and provides facilities for performing basic tasks in the systems. But there exist various instances of the process and module models, as well as there exist many instances of use cases, class, statechart, and sequence diagrams.

(c) seven process models are created and each is refined for a subsystem in the subsystem model.
(d) eleven module models are created and each is refined for a process in the process models.

Moreover, the artefacts of each product member are created. For example, a use case is used to elaborate the satisfaction of the functionalities for each product member. As below, the list of artefacts created for each product member is shown.

**PM1**
(a) four use case descriptions
(b) a class diagram
(c) a statechart diagram
(d) four sequence diagram
(e) source code

**PM2**
(a) four use case descriptions
(b) a class diagram
(c) a statechart diagram
(d) four sequence diagram
(e) source code

**PM3**
(a) four use case descriptions
(b) a class diagram
(c) a statechart diagram
(d) six sequence diagram
(e) source code

*B. Maintenance Phase*

According to software product line-based systems, new requirements management can be facilitated by the identification and analysis of commonality and variability principles among software product line and product members. In particular, the software artefacts are reusable and adaptable. A number of relations between artefacts are detected in order to determine the association between the new requirements and existing software artefacts in product member PM1 and software product line. Different types of traceability relations are created to identify the role of those relations [8]. For example, the relations between the new requirements and software product line; between the new requirements and product member PM1, and between software product line and product member PM1. For instance, there are (a) four use case documents for PM1 and three processes in a process model of software product line that are related in terms of three different types of traceability relations (i.e. satisfies, implements, and refines); (b) one class diagram and four sequence diagrams of software product line that are related in terms of containment. Those relations are then used in new requirements management process.

## 4.3 Waterfall-based Projects

*A. Development Phase*

Similarly, the projects have been developed based on study, analysis, and discussions of business domain. The developers are required to reproduce the software systems based on the same set of requirements. Otherwise, this time they followed the waterfall software process model. According to the waterfall model, a number of artefacts for each single software product are created during software development process. As below, the artefacts of each single software product are checked and submitted to the repository.

**PM1**
(a) a usecase diagram
(b) four use case descriptions
(c) a class diagram
(d) a statechart diagram
(e) four sequence diagram
(f) source code

**PM2**
(a) a usecase diagram
(b) four use case descriptions
(c) a class diagram
(d) a statechart diagram
(e) four sequence diagram
(f) source code

**PM3**
(a) a usecase diagram
(b) four use case descriptions
(c) a class diagram
(d) a statechart diagram
(e) four sequence diagram
(f) source code

*B. Maintenance Phase*

For new requirements management on waterfall-based systems, developers divided their work based on their roles. Firstly, the developers summarized all new requirements from customers and reproduced new user requirement specification. Next, they redesigned the system architecture, components and data models. They applied use case descriptions and use case diagrams to explain the new requirements of the software product. They updated class diagrams, sequence diagrams and activity diagrams of the entire project in this stage. They re-implemented the software by following the documents and used unit tests regularly. When completing all the components, the developers integrated all the pieces together again and began an integration test. Finally, the developer delivered the customers the complete software when all of these stages finished.

## 5. Results and Discussion

In this section, we analyse and evaluate the experiments by focusing on two aspects of measurement: (a) qualitative and (b) quantitative measurement.

### *A. Qualitative Measurement*

In general, qualitative methods and tools for system analysis can address the problem of how to empirically determine the context of software process. In this research, we focused on comparison between two software process methodologies how they are practiced. As mentioned, we have conducted the survey and interview. It has been observed that the customers are satisfied with the software product line resulting projects and teamwork. Moreover, the software product line developers satisfied the process that emphasis the software more than the documentation. However, it has been also noticed that it is easier to train waterfall-based practices to inexperience developers but some experience developers tend to resist some software product line practices because (a) they have to change their style in working, and (b) it costs them for establishing the software product line artefacts.

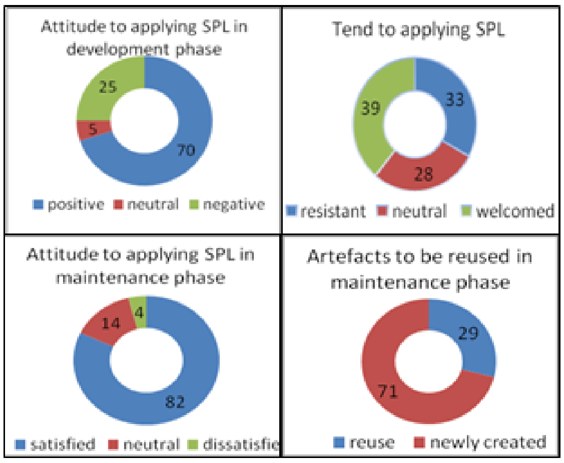

**Figure 4:** Qualitative measurement

According to the survey, it is found that 33% of developers tend to resist software product line practices with the above reasons, whereas 70% of developers are positive to using software product line practices. Particularly, 82% of developers are satisfied when performed the maintenance phase with software product line. Some of software product line artefacts are used during the maintenance phase. And it is satisfied by the developers. However, application engineering process depends on developer' skill. Moreover, the waterfall-based developers are unsatisfied to frequently update the documentation.

### *B. Quantitative Measurement*

Basically quantitative metrics are fundamentally limited to the measurement of the size of system, time and effort spent during software development process. In this research, we measured the total of work hour spent during development and maintenance phases. The results show that the effort metric of software product line-based projects is less than waterfall-based projects. Software product line-based projects enhance the productivity by using existing software artefacts. The methodology supports software reuse at the largest level of granularity.

**Table2 shows the effort spent during software development for each project**

| | Work hour |
|---|---|
| Domain Engineering | 620 |
| SPL-based project1 | 315 |
| SPL-based project2 | 240 |
| SPL-based project3 | 215 |
| Waterfall-based project1 | 465 |
| Waterfall-based project2 | 548 |
| Waterfall-based project3 | 384 |

However, developers spent time and effort to establish domain artefacts. Also, some defects are discovered during the integration process for a product member. It took some effort to fix them. On the other hand, in the waterfall-based team, customers are involved at the inception of project determined requirements and contractual agreement. Developers wrote all documents before coding. Then customers changed some requirements, maybe after they acquired finally product, developers needed to significantly redesign and edit their documents. This took a lot of effort to achieve the task.

For maintenance phase, we measured the total of time to achieve the new requirements. The result show that the spending time of software product line -based projects is less than of waterfall-based projects. Developers who performed the maintenance phase found that well documentation can be useful and reduce the cost to complete the task. In particular, the artefacts of a waterfall-based project are more documentation than a software product line-based project. Otherwise, entire documentation of waterfall process is inaccessible to maintainers whereas documentation of software product line process is restored as repository to support in maintenance and reuse process.

**Table 3 shows the effort spend during the change process**

| | Work hour |
|---|---|
| Software product line -based project1 | 305 |
| Waterfall-based project1 | 380 |

## 6. Conclusions and Future Work

To conclude, we evaluated comparative study between software product line-based process and waterfall-based process. The productivity during development using software product line is higher than that using waterfall-based model. Also, a software product line-based project is more maintainable than waterfall-based one. However, software product line is unsuitable for all projects. It serves the reuse practice in an organization having a large number of products, which have similar requirements and some differences. Developers must consider the characteristics of the project to ensure software product line is appropriate. In the other hand, waterfall process is suitable to serve a software project which is small and has solid requirements.

In the future work, we plan to gather more data from the projects in order to develop statistic evaluation of comparison between two software process models. Additionally, we plan to develop the tools which support the software process. The techniques and approaches for software product line development should be further extended to allow establishing software product line for small- and medium- sized companies. Moreover, an approach to evolve software product line should be investigated in order to enforce a standardized approach for evolution.

## References

[1] Atkinson, C., J. Bayer, and D. Muthig. 2000. *Component-based product line development: The KobrA approach*. Pages 289-310. the 1st Software Product Line Conference, SPLC. Kluwer, Denver, Colorado, USA.

[2] Bayer, J., O. Flege, P. Knauber, R. Laqua, D. Muthig, K. Schmid, T. Widen, and J.-M. DeBaud. 1999. *PuLSE: A methodology to develop software product lines*. Pages 122-131. the Fifth ACM SIGSOFT Symposium on Software Reusability (SSR'99), Los Angeles, CA, USA.

[3] Boehm, B. 1991. *Software Risk Management: Principles and Practices*. IEEE Software, no.pp. 32-41, January.

[4] Clements, P., and L. Northrop. 2004. *A Framework for Software Product Lines Practice*. http://www.sei.cmu.edu/productlines/framework.html

[5] Fenton, Norman E.. 1991. *Software Metrics: A Rigorous Approach*. Chapman & Hall, Ltd., London, UK.

[6] Griss, M. L., J. Favaro, and M. d. Alessandro. 1998. *Integrating feature modeling with the RSEB*. Pages 76-85 in P. Devanbu and J. Poulin, eds. the 5th International Conference on Software Reuse. IEEE Computer Society Press.

[7] Jirapanthong, W. 2008. *An Approach to Software Artefact Specification for Supporting Product Line System*s. the 2008 International Conference on Software Engineering Research and Practice (SERP'08), Las Vegas, Nevada, USA, 2008.

[8] Jirapanthong, W., and A. Zisman. 2009. *XTraQue: traceability for product line systems*. Software and System Modeling 8(1): 117-144 (2009).

[9] Kang, K., S. Cohen, J. Hess, W. Novak, and A. Peterson. 1990. *Feature-Oriented Domain Analysis (FODA) Feasibility Study*. Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA.

[10] Kang, K. C., S. Kim, J. Lee, K. Kim, E. Shin, and M. Huh. 1998. *FORM: a feature-oriented reuse method with domain-specific architectures*. Annals of Software Engineering 5: 143-168.

[11] Linden, F. v. d., J. Bosch, E. Kamsties, K. K¨ans¨al¨a, and H. Obbink. 2004. *Software Product Family Evaluation*. Pages 110-129. the Third International Software Product Line Conference, SPLC 2004. Springer Boston, MA, USA.

[12] Mills, Everald E. 1998. *Software Metrics*. SEI Curriculum Module SEI-CM-12-1.1, December.

[13] Sharpe, J.L., and J. W. Cangussu. 2005. *A productivity metric based on statistical pattern recognition*. Computer Software and Applications Conference, 29th Annual International, 26-28 July.

[14] Weiss, D. 1995. *Software Synthesis: The FAST Process*. the International Conference on Computing in High Energy Physics (CHEP), Rio de Janeiro, Brazil.