Proceedings

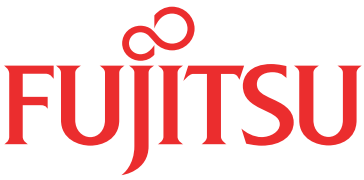
The First International Workshop on Management and Economics of Software Product Lines (MESPUL 07)

Sponsored by

Information Processing Society of Japan (IPSJ),
Special Interest Group on Software Engineering (SIG-SE)

In cooperation with
The Institute of Electronics, Information and Communication Engineers (IEICE)
Special Interest Group on Software Science (SIG-SS)
The Institute of Electronics, Information and Communication Engineers (IEICE)
Special Interest Group on Knowledge-based Software Engineering (SIG-KBSE)
Japan Society for Software Science and Technology (JSSST), FOSE


Microsoft


FUJITSU


HITACHI
Inspire the Next


Google


UNIVERSITAS NANZAN
N
ZAN
N
HOMINIS DIGNITATI

Proceedings

The First International Workshop on Management and Economics of Software Product Lines (MESPUL 07)

Nagoya, Japan, December 3, 2007
Co-located with the 14th Asia-Pacific Software Engineering Conference (APSEC2007)

Workshop Organizers:

Muhammad Ali Babar (Lero, University of Limerick, Ireland)

Makoto Nonaka (Toyo University, Japan)

Table of Contents

Keynote Presentation

Presentation

Economic Decision-Making for Architectures

Rick Kazman
Department of Information Technology Management
Shidler College of Business
University of Hawaii

Abstract

There is often a lack of connection between how executives and managers define and foresee value and how architects enable or disable those value propositions through their design decisions. This lack of effective communication causes the partnership between architects and executives to be a weak one. Opportunities are often missed where the architect may not only serve, but also inform executives on value-driven design decisions. This information exchange is particularly critical when the business needs to deal with architecture evolution and uncertainty. Currently architects and project managers simply do not have the tools to rigorously consider the implications of software architecture decisions with respect to multiple quality attributes, and with respect to potentially uncertain future market conditions. In this talk I will discuss two methods for improving the analysis of (and communication of) economic issues related to architectures: the Cost-Benefit Analysis

Method (CBAM), which was developed to guide an architect in making software architecture choices; and Architecture Planning for Evolution, which takes CBAM-like reasoning and applies it to a time-series of envisioned changes, as a means of planning for and optimizing the choices over a long time period.

Core Asset Scoping Method: Product Line Positioning Based on Levels of Coverage and Consistency

Mari Inoki
Toshiba Solutions Corporation and Waseda University
inoki.mari@toshiba-sol.co.jp

Yoshiaki Fukazawa
School of Science and Engineering Waseda University
fukazawa@waseda.jp

Abstract

Core asset scoping is an essential activity that has economic impact on product-line-driven software development. For an enterprise that has several product lines, it is necessary to correctly select product lines in which the enterprise invests before scoping individual product lines. In this paper, we propose a core asset scoping method on the basis of two metrics: the levels of coverage and consistency of core assets. The method makes it possible to represent the scope of all core assets and clarify the positions of product lines in the enterprise.

1. Introduction

1.1. Importance of scoping in software product line engineering

Software reuse helps in the development of software at a lower cost, with higher quality and in a shorter time by accumulating and reusing software parts, the quality of which is guaranteed [1]. The form of reusable artifacts has been changed from source codes in some programming languages to, for example, components and application frameworks. In line with such a change, software product line engineering has emerged. In the present paradigm, software reuse is facilitated by focusing on various kinds of artifacts, although it is limited to the development of similar products in the same domain [2-3]. In a software product line engineering paradigm, an organization draws a product roadmap, prepares software assets, which are called core assets, and then develops software products by reusing core assets [4].

An economic benefit in product line development is estimated according to the differences between the cost of product development by reusing core assets and the investment cost for developing and maintaining core assets [5]. An optimal investment is indispensable for economic benefit. Scoping, that is, deciding what core assets should include is the key to success in product line engineering [4]. If the scope of core assets is too wide, the enterprise will develop core assets that will not be reused for developing products resulting in extra costs and creating a negative impact on returns on investment. On the other hand, if the scope is too narrow, core assets will not include all intersection parts common to several products; the enterprise will need to repeat the development of such parts every time it develops new products [6].

Considerable research effort on core asset scoping has been presented so far. Schmid surveyed about thirty techniques for scope definition and clarified those features [7]. PuLSE-Eco [8] and a scoping method based on an object-oriented approach [9] are typical examples of core asset scoping methods. However, considering that the target domains of product lines and the technologies used to develop, reuse and maintain them are diverse, much still remains to be done.

1.2. Problem of scope definition in enterprise

We have confronted the following problems in application examples that are used to decide plans for the investment and development for product lines in an enterprise. However, these have not been addressed by the scoping methods that have been proposed so far.

Problem 1: There is no method for evaluating product lines without depending on domains.

An enterprise that has several product lines needs to determine the product lines in which the enterprise should invest. Target domains for these product lines are diverse. Therefore, it is important for the enterprise to compare product lines without depending on the features of the products or the technologies to be used. However, such a method has not been developed so far.

Problem 2: Simple and easy-to-understand methods have not been developed.

When scoping core assets, it is important to take various opinions into account; such opinions are

generally obtained from not only top executives but also people in marketing and development departments. However, if the enterprise is undecided on whether to invest in the target product line, it is difficult to spend significant costs on scoping activities.

A scoping method that is simple and easy to use for diverse stakeholders and does not cost much to apply is necessary. However, such a method has not been proposed so far.

Problem 3: Quality for all core assets is not considered.

Let us take an example of core assets for a domain including analysis, design, test and implementation models. If individual models are verified and validated on the basis of the test model, such core assets are favorable from the viewpoint of quality. However, its cost is more than that in the case without the verification and validation of the models. Therefore, the enterprise should focus on not only the quantity but also the quality of the core assets; the investment cost in product lines should include the costs of improving the quality of all core assets. In the research field of product line scoping, metrics for measuring both quantity and quality of all core assets have not been provided.

1.3. Research approach

In this paper, we propose a core asset scoping method. The objectives of our method are to solve the problems described in 1.2 and to help the enterprise plan its investment in product lines so as to contribute to the optimization of product-line-driven software development. Our method includes the following features.

(a) Provide metrics for measuring the scope of all core assets in terms of quantity and quality

(b) Define the metrics that do not depend on domain-specific issues including the product features and technologies to be used

(c) Represent situations of core assets that can be easily understood for diverse stakeholders using the metrics

(d) Assist the enterprise in comparing several product lines and determining in which product lines to invest

2. Metrics and Types

In this paper, we propose two metrics: level of coverage (LCV) and level of consistency (LCC).

2.1. Metrics for scoping: LCV

LCV is a quantitative measure that describes to what degree elements of core assets are prepared. Considering returns on investment (ROI) analysis, it is impossible for an actual organization to prepare all artifacts developed or used in a software development lifecycle. LCV should be adjusted depending on the conditions of an organization. Core assets with a high LCV are substantial and sufficient for developing various products by reusing core assets.

The equation of LCV is shown as (s). LCV is derived from the ratio of Ns to Nd, where Nd is the number of items in a coverage checklist (CCL) expected to pass and Ns is the number of items in the CCL defined in the organization as standards.

$$LCV = Nd / Ns \qquad (s)$$

CCL is used to check to what degree common and variable elements of core assets are prepared; items in the checklist are defined by all kinds of artifact.

Assuming that there are two kinds of artifact, (a) and (b), in an artifact catalogue defined as organization standards, an example of CCL is shown as follows.

(a) Structural diagram of a conceptual data model

(a1) Are all data common to the target domain exhaustively defined in the diagram as entities?

(a2) Are all data specific to main product models defined in the diagram as entities?

(a3) Are all data specific to the products that have been produced or have not been decided to be sold defined in the diagram as entities?

(b) Specification of data definition in a conceptual data model

(b1) Are all data common to the target domain exhaustively defined in the specification of data attribute definition?

(b2) Are all data specific to main product models defined in the specification of data attribute definition?

(b3) Are all data specific to the products that have been produced or have not been decided to be sold defined in the specification of data attribute definition?

If there are two artifacts in the artifact catalogue defined in the standards mentioned in the above example, Ns is 6. By assuming that the target core assets include both (a) and (b), and they satisfy (a1) and (b1), Nd is 2. In this case, LCV equals 2/6, approximately 0.33.

2.2. Metrics for scoping: LCC

LCC is a qualitative measure of all core assets; it describes the consistency of all core assets in a product line. Core assets with a high LCC do not contradict each other.

By assuming that the following two core assets in an order processing system domain includes a design model and an order processing component, core assets

in Case 2 are consistent with each other and have a high LCC.

Case 1: The order processing component is not validated or verified on the basis of the design model; the detailed behavior of the component including the specification of exception handling is specified differently from the actual behavior of the component.

Case 2: The order processing component is validated and verified on the basis of the design model. The order processing component is implemented on the basis of the behavior and structure models specified in the design model; the behaviors of the order processing component correspond to its specification in the design model.

Core assets are not products themselves. Therefore, order processing components in both cases can be core assets. The LCC of core assets affects the cost of product line development. An organization must adjust LCC depending on its conditions.

The equation of LCC is shown as (t). LCC is derived from the ratio of Nq to Np, where Nq is the number of items in a quality inspection checklist (QCL) that are expected to pass quality inspection, and Np is the number of items in the QCL defined in the organization as standards.

$$LCC = Nq / Np \qquad (t)$$

The QCL is used to check the consistency of all core assets in a product line. On the basis of the artifact catalogue defined in the organization as standards, quality inspection points are defined between artifacts defined in the standards.

If there are three kinds of artifact, namely, a design structure model, a design behavior model, and a component, an example of a QCL is shown as follows.

(q1) Are components implemented on the basis of the specification of the design structure model?

(q2) Are components implemented on the basis of the specification of the design behavior model?

(q3) Does every behavior of a component correspond to the specification of the components in the design behavior model?

In the above case, Np is 3. By assuming that Case 1 satisfies none of them, LCC is 0/3 = 0. On the other hand, because Case 2 satisfies (q1), (q2) and (q3), LCC is 3/3 = 1.

2.3. Core asset types

By setting reference LCV and LCC, the conditions of core assets are classified into four types, as shown in Table 1. Comparing product lines using these four types makes it possible to clarify their positions in the enterprise; this is useful for deciding in which product line to invest.

Type A: Treasure: The core assets include a series of artifacts developed from or used in the analysis, design, programming, and test phases of the software development lifecycle; the core assets are consistent with each other. An organization develops core assets of this type when it expects a high ROI and considers a product as being strong in the marketplace.

Type B: Mixture of chaff and grain: The core assets include a series of artifacts developed from or used in the analysis, design, programming, and test phases of the software development lifecycle. However, the core assets are not consistent with each other. An organization develops core assets of this type when it does not expect a high ROI but has already identified several candidate core assets for the target product line.

Type C: Gemstones: The core assets are consistent with each other, although elements are limited. An organization develops core assets of this type when it is necessary to enter a new marketplace quickly.

Type D: Stones: The core assets include minimum artifacts, for example, software components and design documents; they are inconsistent. An organization develops core assets of this type when the barrier for entry into the marketplace is high and it is uncertain whether to consider a product as being strong in the marketplace.

Table 1. Types of core asset.

Classified into two groups by comparison with reference value

LCV \ LCC	Consistent Core Assets	Inconsistent Core Assets
Full Core Assets	Type A: Treasure	Type B: Mixture of Chaff & Grain
Partial Core Assets	Type C: Gemstones	Type D: Stones

3. Core asset scoping process

Assuming that there are following three layers of organization in the enterprise, we propose a core asset scoping processes.

The top management layer includes executives of the enterprise. They analyze and examine product line development plans from individual business units and determine in which product lines to invest, and the amount and period of investment.

The business unit layer is in charge of business for a specific business domain. Each business unit (hereafter, called BU) creates a plan for product line development and submits it to the top management layer. When the submitted product line plan is accepted by the top management layer, BU begins to develop the product line and maintains it.

The cross-BU management layer is in charge of developing and maintaining organizational standards that are used by all BUs under the control of the top management layer.

The core asset scoping processes consist of three parts which correspond to the above layers. An overview of the processes is shown in Figure 1. A detailed definition of individual processes is as follows.

(a) Processes for top management.

(a1) Determine strategies. The enterprise defines its goal, and creates a road map to achieve the goal.

(a2) Judge submission from BUs. The top management layer examines product line development plans submitted by individual BUs, and determines in which product lines to invest, and the amount and period of investment. When the top management layer compares several product lines submitted by BUs, it utilizes LCV, LCC and the types of core asset.

(a3) Perform investment. The top management layer invests on the basis of the decision in (a2).

(a4) Analyze results. The top management layer examines the progress of the product-line-driven software development in BUs, and changes the amount and period of investment, if necessary.

(b) Processes for BU

(b1) Determine strategies for BU. On the basis of the enterprise's strategies, the BU defines a goal of the BU and creates a road map to achieve the goal.

(b2) Define the scope of the product line. On the basis of the strategies of the BU, the scope of the target product line is defined by the following steps.

(b21) Draw a product road map. The BU analyzes the target domain and draws a product road map to achieve the goal.

(b22) Define the scope of the target product line from the viewpoint of developers. The BU derives the LCV and LCV, and specifies the current and goal types of the target core asset.

(b23) Analyze ROI. The BU estimates the development cost of core assets and cost savings obtained from product-line-driven development.

(b24) Plan product line development. The result defined in the step (b2) is submitted to the top management layer.

(b3) Develop and reuse core assets. When the investment is performed by the top management layer, the BU starts to develop core assets, develops products by reusing core assets, and maintains core assets on the basis of the product line development and reuse plan.

(c) Processes for cross-BU management

(c1) Define the organization's standards. The organization that crosses BUs defines the organization's standards; they include an artifact catalogue, reference values for LCV and LCC, CCL and QCL, and equations for ROI.

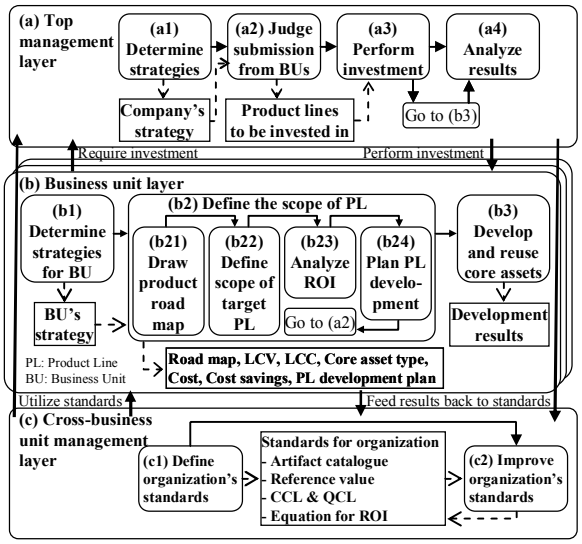


Figure 1. Core asset scoping steps.

(c2) Improve the organization's standards. The organization analyzes the results of the scope definition defined by individual BUs and the investment by the top executive, and improves the standards by feeding know-how back to the standards.

4. Application example

In this section, an application example of our scoping method is shown, assuming that an enterprise that plans to have several product lines are now trying to decide in which product lines to invest. We generated the data and information shown in this section by interviewing stakeholders relevant to the actual product line development.

4.1. Target business units

A top management layer is trying to decide in which product lines to invest by examining and comparing product line plans submitted by individual BUs. The total budget of investment has already been defined. The target product line candidates are as follows.

PL1: product information traceability systems

Traceability is the ability of product information to be traced along the distribution chain. Product information includes the locations and arrival time, departure time, and status of products and how products are packaged. The product information traceability system processes such information. Users refer to the product information accumulated in the system and utilize it for checking the safety of the products. The BU enters this marketplace for the first time. There are two kinds of product model: a model

for large order-made products and a model for small mass-producted items. The former traces products individually, and the latter traces not only individual products but also containers in which individual small products are packaged.

PL2: editing data support systems

An editing data support system helps an editor edit data; the system transforms data to another representation and deploys that data in an optimal place on a computer screen or paper.

The BU entered that marketplace more than twenty years ago. To meet the requirements of several customers, the core assets include many variations. Although the organization maintains core assets for a long time, some problems occur. For example, there are core assets that have not been used recently. Core assets including unnecessary ones entail cost for product validation. In addition, core assets that are similar to each other prevent a product developer from choosing appropriate core assets smoothly. The BU is trying to reorganize core assets so as to solve the above problems.

PL3 : business operation processing systems

The system assists users of an office for a specific business field. The target domain is very large; the BU enters this marketplace for the first time. There are two product models: models for huge and small organizations.

4.2. Results

We applied our core asset scoping method to the analysis of the above three product line candidates.

4.2.1. Definition of organization standards. The organization standards prepared include the following.

- Artifact catalogue: a catalogue of 59 kinds of artifact
- CCL: a checklist of 177 items
- QCL: a checklist of 96 items
- Reference LCV and LCC were both set to be 0.5

4.2.2. Scoping core assets by BU. Individual BUs calculated LCV and LCC and derived the types of the target product lines, and submitted their product line development plans to the top management layer.

The cost of implementing core assets was also estimated using an ROI method such as Böckle et al.'s method [5].

4.2.3. Scoping product lines by top management layer. The results of the calculation of LCV, LCC and types that were submitted to the top management layer are shown in Table 2 and Figure 2. In Table 2, "*1" means the rank of the amount of investment applied by BUs. "*2" means the period of time until ROI becomes positive. As shown in Table 2, among three product lines, the amount of investment and cost savings of PL2 estimated are No.1. The cost savings of all product lines are estimated to become positive within three years.

Table 2. Example of LCV and LCC.

ID	LCV	LCC	Type	Rank (*1)	Period (*2) (Year)	ROI Rank
PL1	0.28	0.65	TypeC	3	3	3
PL2	0.88	0.20	TypeB	1	3	1
PL3	0.75	0.99	TypeA	2	3	2

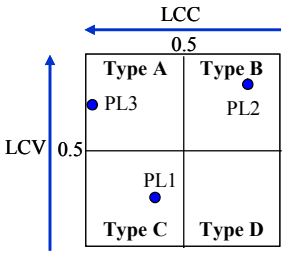


Figure 2. Example for comparing 3 product lines.

4.2.4 Comparison with actual investment results. Actually, the investment in PL1 was executed because the development plan was considered to be appropriate, although the amount of cost savings was low. The investment in PL2 was not executed, because the amount of investment was large and the risk seemed to be high, although PL2 was expected to have a high ROI. Instead of executing the investment, the top management layer instructed the BU to review PL2's development plan, and determine if it is effective to refactor existing core assets. Regarding PL3, it is a large-scale domain in which the BU will enter for the first time. The top management layer determined to execute the investment in PL3 so as to make PL3 a strong point of the enterprise.

The types derived from LCV and LCC match the actual decisions of the top management layer. A detailed explanation is as follows.

(1) The target domain of PL1 is a small-scale domain that the BU enters for the first time. However, the BU values the quality of the product line. This situation is made visible as type C, and the top management layer evaluated that it is appropriate to invest in PL1.

(2) Regarding PL2, the BU has entered the target domain already and has extensive experience in developing products in the domain. However, the BU did not consider the quality of the product line. This situation is made visible as type B, and the top management layer decided that it is necessary to reconsider to invest in PL2.

(3) PL3 is a domain assumed to be a strong point of the enterprise. The BU values both the quality and quantity

of the product line. This situation is made visible as type A, and the top management layer decided that it is favorable to invest in PL3.

5. Discussion

We have evaluated the advantage of our core asset scoping method by considering whether it gives valid solutions to the problems described in 1.2.

(1) Can our method evaluate product lines without depending on domain specific information?

LCV is derived from the numbers of items in CCL and items that are expected to pass CCL; LCC is derived basically from the numbers of items in the QCL and items that are expected to pass the QCL. These are independent from the target domains and technologies to be used. Therefore, LCV and LCC can be calculated regardless of specific domains.

The LCV and LCC range from 0 to 1; the core asset type is derived from these values. Mapping product lines into a map divided into the four types shown in Table 1 makes it possible to clarify the positions of the product lines in the enterprise. The process of investment judgment, which has been tacitly performed by the top management layer, can be made visible by calculating LCV and LCC and deriving the types of core assets. Our method is useful to the enterprise for comparing product lines and determining in which product lines to invest.

(2) Is our method simple and easy to understand?

LCV and LCC can be measured using checklists and are easy to calculate. If diverse stakeholders use these checklists and calculate LCV and LCC from individual standpoints, it will help them to communicate with each other and to define the scope of core assets smoothly.

(3) Does our method consider the quality of all core assets?

LCC represents the quality policy for the target product line. The comparison of LCC of the target product line with the reference LCC yields two choices for the product line: to consider or not consider quality of core assets. In this way, our method considers quality of all core assets.

Up to now, there have been no methods available for directly representing the consistency between core assets. Therefore, the enterprise tended to ignore the costs of improving the quality of all core assets. However, in our method, the relationships showing the consistency between core assets are represented by LCC; this helps the enterprise not only to recognize how much effort is necessary to maintain consistency between core assets but also to clarify the verification and validation policy before developing core assets.

(4) Issues not solved by our method

An artifact catalogue should be defined and shared in an enterprise as standards. For the enterprise that has several product lines for several domains, an artifact catalogue may not be sufficient; several catalogues may be necessary depending on target domains.

The reference LCV and LCC, and CCL and QCL need to be defined in standards of the enterprise. To improve the precision of scoping core assets, we will continuously improve the standards and our method on the basis of feedback from both the actual development and reuse of core assets.

6. Conclusion

In this paper, we proposed a core asset scoping method on the basis of two metrics: LCV and LCC. Our method specifies the core asset situation using LCV and LCC and the core asset types obtained from calculating both values. We applied the method to the analysis of three product line candidates. The application results showed that the types derived from LCV and LCC match the actual decision of the top management layer; our method makes it possible to clarify the positions of product lines in an enterprise and helps the enterprise to determine in which product lines to invest.

7. References

[1] H. Mili, et al., *Reuse-Based Software Engineering Techniques, Organization, and Controls*, Wiley & Sons, 2002.

[2] A. Birk et al., "Product Line Engineering: The State of the Practice", *IEEE Software*, *Vol. 20, No .6*, 2003, pp. 52-60.

[3] V. Sugumaran, et al. "Software Product Line Engineering", *Communications of the ACM, vol. 49, No. 12*, 2006, pp. 29-32

[4] P. Clements, and L. Northrop, *Software Product Lines: Practice and Patterns*, Addison-Wesley, 2001.

[5] G. Böckle et al., "Calculating ROI for Software Product Lines", *IEEE Software, Vol. 21, No. 3,* 2004, pp. 23-31.

[6] K. Schmid, "A Comprehensive Product Line Scoping Approach and Its Validation", *Proc. of the 24th International Conference on Software Engineering*, 2002, pp. 593-603.

[7] K. Schmid, "Scoping Software Product Lines, An Analysis of an Emerging Technology", *Proc. of the First Software Product Line Conference (SPLC1)*, 2000, pp. 513-532.

[8] J. DeBaud and K. Schmid, "A Systematic Approach to Derive the Scope of Software Product Lines", *Proc. of the 21st International Conference on Software Engineering*, 1999, pp. 34 -43.

[9] S. Cohen and L. Northrop, "Object-Oriented Technology and Domain Analysis", *Proc. of the Fifth International Conference on Software Reuse*, 1998, pp. 86-93.

A Simulation Study for Predicting Cost and Schedule Overruns

Makoto Nonaka
Faculty of Business Administration
Toyo University, Japan

Abstract

Software product line development has been demonstrated as a promising approach to shorten time-to-market by achieving large-scale strategic reuse. An important longer-term issue from the viewpoint of management and economics of a software product line is the efficiency of investments made in the product line over its life. There are a wide range of investment targets for the product line, but its architecture and quality improvement can be considered as important investment targets.

Architecture investment can bring benefits in reducing implementation effort for new functionality, and in shortening time-to-market. Architecture investment is realized as funding for activities to increase the level of reuse in the product line architecture and prevent architecture degradation. The reusability and flexibility of product line architecture should be maintained or improved, as it impacts development activities throughout the life of the product line.

Quality improvement of a product line requires a better process, and a better process requires the application of improved software engineering practices. Process improvement can lead to a lower total cost of software quality overall, but often takes additional cost for each individual activity. Substantial effort on these activities will be required to achieve extremely high reliability.

Investments in both product line architecture and process improvement can have a great impact on reducing unplanned work. However, over-investment can sometimes makes the total cost of quality worse than expected. The impact of architecture investment will change depending on how many products are planned to be developed. A development plan should take into account the optimum level of investment for the target product quality level, the expected lifetime of the product line, and the level of uncertainty in demand for the products.

In this presentation, we introduce a stochastic simulation model[1] for estimating additional effort of unplanned work in product line development. The model includes considerations of: uncertainty of unplanned work caused by requirements change and residual defects, the level of risk of estimated additional effort derived from uncertainty, concurrency among core projects and product projects. We focus on architecture investment in terms of the degree of reuse and inter-asset dependency, and quality from process improvement in terms of defect injection and detection rates. We evaluate several scenarios through simulation from the viewpoint of lifetime impacts of these investments.

References

[1] Nonaka, M., Zhu, L., Ali Babar, M. and Staples, M., Impacts of Architecture and Quality Investment in Software Product Line Development, *Proc. 11th International Software Product Line Conference (SPLC 2007)*, pp.63-73, 2007.

ISBN 978-4-915256-68-4 C3040
Publisher :
Information Processing Society of Japan
Kagaku-kaikan (Chemistry Hall) 4F
1-5 Kanda-Surugadai, Chiyoda-ku, Tokyo 101-0062 JAPAN
Tel:+81-3-3518-8374 Fax:+81-3-3518-8375