Continuous Quality Assessment for Software Product Lines

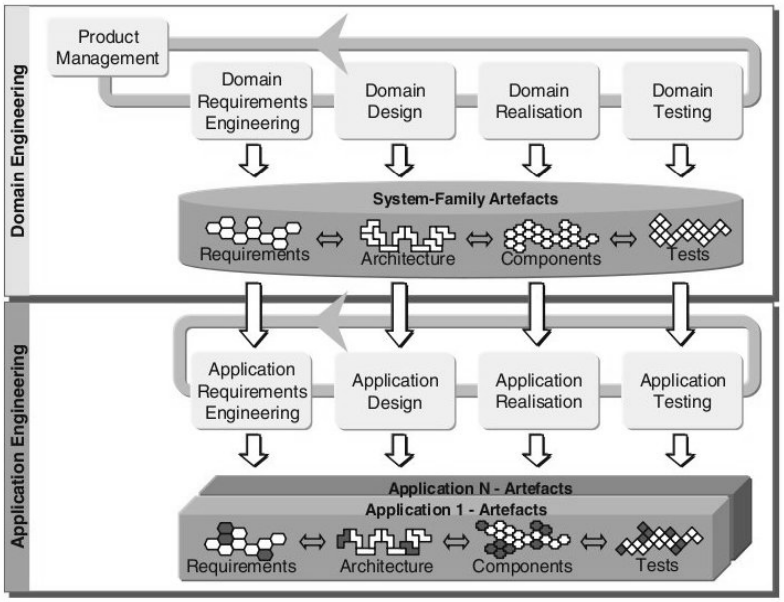
Francisco Aragón
Cuauhtémoc Lemus

CIMAT, A. C. - Unidad Zacatecas
Blvd. L. del Patrocinio #102
Fraccionamiento Lomas del Patrocinio
C.P. 98070
Zacatecas, Zacatecas, México
+52 492 998 1529

What is a Software Product Line?

An SPL is a set of software-intensive systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way [Clements 2002].

The two-life-cycle model of software product line engineering



CIMAT

The problem and its justification 1/4

Software systems are constructed to realize business or mission goals [Kazman 2005].

How do we assure the products in a software product line help to accomplish the intended business goals?

CIMAT

The problem and its justification 2/4

Non-functional requirements also known as quality attributes such as the stakeholders' expected performance, security, modifiability and availability among others distinguish a product that merely does what it is supposed to do from one that delights its customers [Weigers 2003].

Most existing requirements models and requirements specification languages lack a proper treatment of quality characteristics [Chung 2009].

The problem and its justification 3/4

Quality attribute requirements drive architectures, and business goals drive quality attribute requirements [Clements 2010].

Software architecture is the place where nearly all of the system's quality attributes are allowed or precluded [Clements 2003].

The problem and its justification 4/4

Software architecture is the bridge between the business goals and the realized system [Kazman 2005].

The quality of a software system is spread all over the artifacts that comprise a product and has to be assured along the development.

Related and prior work

- For Business Goals elicitation
  - Classification of business goals, Business Goal Scenarios, PALM
- For Quality Attributes specification
  - Quality Scenarios
- For Architectural Elements definition
  - Tactics, Styles and Patterns
- For Code Elements compliance
  - Static and dynamic architecture compliance checking
- For Testing Elements
  - Test cases

The claim

A practical way for a continuous quality assessment would provide the means to guide the quality decision-making process, to improve the communication of quality issues and to trace quality along the SPL development assuring products help satisfy business goals.

Research Question 1/2

How can continuous quality assessment be performed along the software product line development supporting business goals' achievement?

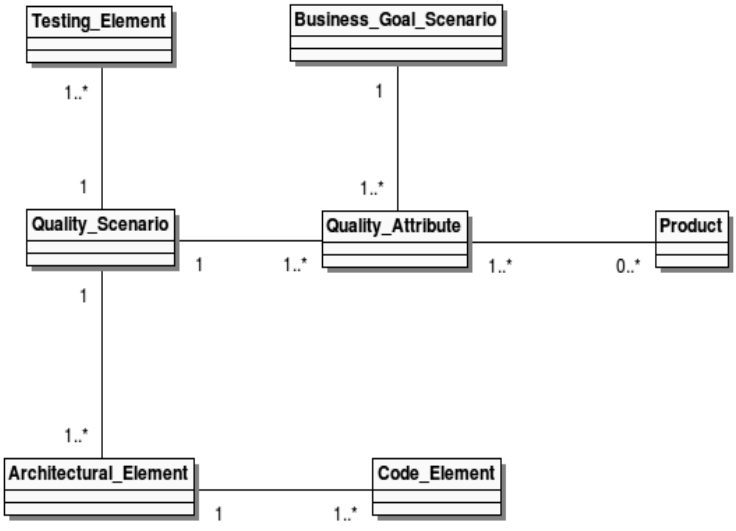
What is continuous quality assessment?
Does continuous quality assessment exists?
Is continuous quality assessment the same as quality aware SPL engineering? If not how they differ?
What I call continuous quality assessment differs of quality management? And if so, how it differs?

Research Question 2/2

If it exists, what's the purpose of continuous quality assessment?
Is continuous quality assessment something that someone actually does for software product line development?
How can software quality be measured?
What types of software quality are present during software product line development?

CIMAT

Where is the quality?



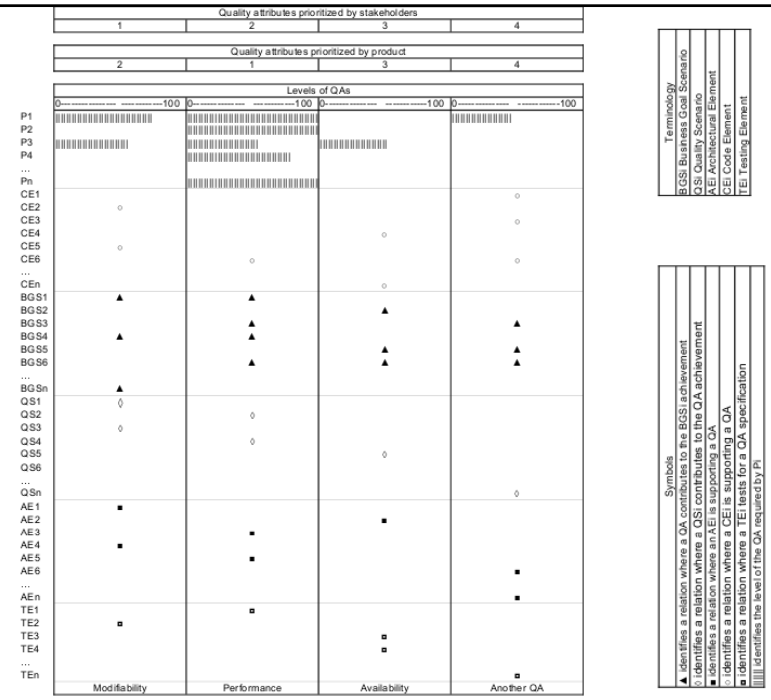
CIMAT

A Partial Solution

Make quality documentation explicit, tracing quality along the Software Product Line development using an Orthogonal Quality Model.

CIMAT

The Orthogonal Quality Model



CIMAT

The research expected contributions

During software product line engineering we will count with:

1. A model to trace quality
2. Continuous quality assessment procedures

Progress in solving the stated problem 1/3

So far we have tried to build the Orthogonal Quality Model from a set of available assets of the Arcade Game Maker Pedagogical Product Line to identify quality assessment tasks performed during SPL development.

The Arcade Game Maker (AGM) product line is an example product line created to support learning about and experimenting with software product lines. The product line encompasses three simple arcade games to create nine products.

Progress in solving the stated problem 2/3

It was possible to identify the Business Goals and specify them through business goals scenarios.
We identified the quality attributes required by the products, but have not done the exercise to apply PALM and check if they can be derived from the business goals.
Quality attributes were specified in quality scenarios.
We found test cases for the performance quality attribute.

Progress in solving the stated problem 3/3

The architectural elements where defined from the selection of a modified MVC pattern. But there is no explicit evidence of the application of tactics.
We have not done the exercise to try to derive the architectural elements from the quality scenarios using tactics, styles and patterns.
The coded elements, commons and specific ones, are not available. We will request them to check compliance between the planned architecture and the implemented architecture.

The method

Define a Case Study choosing a limited number of quality attributes and a specific domain.

Possible domain: smart spaces.

Evaluate our proposed solution throughout the building of a product line of smart spaces for schools.

References

[Clements 2002]
Clements, P. & Northrop, L. Software Product Lines: Practices and Patterns. Boston, MA: Addison-Wesley, 2002.

[Kazman 2005]
Kazman, R. & Bass, L. Categorizing Business Goals for Software Architectures (CMU/SEI-2005-TR-021). Software Engineering Institute, Carnegie Mellon University, 2005. http://www.sei.cmu.edu/reports/05tr021.pdf

[Weigers 2003]
Weigers, K.: Software Requirements: Practical Techniques for Gathering and Managing Requirements Throughout the Product Development Cycle. Microsoft Press, 2003.

[Chung 2009]
Chung, L. & do Prado Leite, J.C.S.: On Non-Functional Requirements in Software Engineering. Mylopoulos Festschrift, LNCS vol. 5600, pp. 363—379, Springer, Heidelberg, 2009.

References

[Clements 2010]
Clements, P. & Bass, L. Relating Business Goals to Architecturally Significant Requirements for Software Systems (CMU/SEI-2010-TN-018). Software Engineering Institute, Carnegie Mellon University, 2010. http://www.sei.cmu.edu/reports/10tn018.pdf

[Clements 2003]
Clements, P.; Kazman, R.; & Klein, M. Evaluating Software Architectures: Methods and Case Studies. Addison-Wesley, 2003.