An Aspect-Oriented Design Approach for Product Line Architectures

Software Architectures Workshop
June 2009

Karen Cortés Verdín
Agenda

- Background
- AOPLA
- Conclusions
Background

What is a SPL (Software Product Line)?

A Software Product Line 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.”

Problems addressed by SPLE

- Disatisfaction with current project/product performance
- Need to reduce cost and schedule
- Complexity of managing and maintaining too many products variants.
- Lack of staff
- Need to quickly respond to customer/marketplace demands.
What is SPL/E about?

- What is SPL/E about?
  - SEI’s framework for PLP

ESAPS & CAFÉ Product Family Engineering Process
SPL main characteristics:

- Two-tier organization
- Planned and proactive reuse of core assets
- Architecture-centric development
Issues in PLA

- Definition of a reference architecture
- Products quality attributes as well as PL-specific quality attributes
- Support for PL evolution
- Commonality and variability
AOSD

- Applies the principle of separation of concerns for obtaining software that is easier to evolve, maintain, comprehend, customize and reuse.

- AOSD methods encompass the means for the identification, modularization, representation and composition of crosscutting concerns.
Crosscutting concerns or aspects are “those concerns that cannot be expressed as separate modules . . . resulting in implementations scattered over multiple operations.”

An Early Aspect is a concern that crosscuts a requirements’ or architecture’s artifact.

Example: Personnel system

- The personnel feature manages basic information about employees
- The payroll feature maintains salary and tax information

Figure 1. Employee class hierarchy. Underlined methods belong to the Payroll feature, while others belong to the Personnel feature.
- Each kind of employee is a data concern.
- The OO paradigm allows the data concerns to be encapsulated within classes, so that all the software associated with each data concern can be localized.
The other kinds of concerns cannot be represented effectively (Tyranny of the dominant decomposition)

The tyranny results in problems related to difficult and costly evolution, low reuse, complicated integration and brittle software
A different organization requires a similar software but with some different requirements:

- They need not maintain payroll information, since a subcontracted organization handles this.
- Business rules are different. For example, an employee only has one manager.

What are the implications of these new requirements?
Aspects and architecture design

- In architecture design, aspects can crosscut several architectural elements

- If such aspects are overlooked during architecture design, they may lead to tangled code, and therefore compromise the system’s specified quality factors
AOPLA

- Aspect-Oriented approach for PLA design, including:
  1. The early identification of concerns
  2. The consideration of concerns during the architecture design process
3. The definition of an architecture model encompassing:

a) PL-specific quality attributes,
b) Products quality attributes
c) commonality and variability support,
d) generality, and
e) aspects
Process

- Business case definition
- Domain engineering
- PL portfolio definition
- Domain analysis
- Domain modeling
- Concern modeling (Cosmos)
SPL use case modeling

SPL Requirements specification

CORE

CAFÉ Quality model

Architecture views and styles selection

Architecture modeling

Evaluation
Portfolio

<table>
<thead>
<tr>
<th>Product 2 (for developers)</th>
<th>Product 1 plus:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2. Security: allowing access only to authorized users and protecting data integrity</td>
</tr>
<tr>
<td></td>
<td>6. Generation of the requirements specification document (reusable)</td>
</tr>
<tr>
<td></td>
<td>18. Support for elaborating, updating and viewing test cases which are reusable</td>
</tr>
<tr>
<td></td>
<td>19. Automatic generation of test cases from the development log</td>
</tr>
<tr>
<td></td>
<td>20. Calculation, analysis and re-calculation of use cases points in an efficient and precise way so that the calculations are reusable. This includes the selection of a context of a project already defined within the CASE tool or of a project outside the tool.</td>
</tr>
<tr>
<td></td>
<td>21. Support for elaborating test plans which are reusable</td>
</tr>
<tr>
<td></td>
<td>22. Support for elaborating test procedures in a way that they are reusable</td>
</tr>
<tr>
<td></td>
<td>23. Support for recording the results of test cases</td>
</tr>
<tr>
<td></td>
<td>24. Support for the recording and follow-up of defects</td>
</tr>
<tr>
<td></td>
<td>25. Risk management: identification, assessment, monitoring (mitigation plan elaboration and follow-up) and contingency plan elaboration and follow-up.</td>
</tr>
</tbody>
</table>

CIMAT, A.C.
Domain analysis and modeling

- Context model
- ERD
- DFD
- Feature modeling
Concern modeling

- Cosmos is a concern modeling schema that allows the handling of concerns as first-class entities.
- Concerns are . . . “those interests which pertain to the system’s development, its operation or any other aspects that are critical or otherwise important to one or more stakeholders”\(^4\)

---

Concerns arise at every stage of the life cycle, spanning activities, artifacts, methods, and tools.

- Ease of use
- Availability
- Correctness

- Key functionality
- Data integrity
- Security
- Availability

- Requirements
- Correctness
- Understandability

- Ease of evolution
- Ease of modification
- Understandability
- Modularization
- **Functionality**
  - Project
  - Script
  - Track
  - Scene
  - Quintet
  - Dialogue
  - Table
  - Glossary
  - Cost
  - . . .

- **Entity**
  - Project
  - Script
  - Track
  - Scene
  - Quintet
  - Dialogue
  - Table
  - Glossary
  - . . .

- **Feature**
  - Function Points
  - Use Case Points
  - NSN
  - Attitude Survey
  - Glossaries and tables
  - Script management
  - Development Log
  - . . .
- **Properties**
  - Evolvability
  - Generality
  - Derivability
  - Reusability
  - Security
  - Modifiability
  - Portability

- **Topics**
  - Function Points
  - Use Case Points
  - NSN
  - Attitude Survey
  - Glossaries and tables
  - Script management
  - Development Log
  - SOM (Semantic Objects Model)
  - ERM (Entity-Relationship Model)
  - Prototype generation
  - SRS document
  - . . .
  - Due to its crosscutting property:
    - Evolvability
    - Generality
    - Derivability
    - Reusability
    - . . .
CORE (Concern-Oriented Requirements Engineering)

- Concerns from domain engineering are mapped to requirements and use cases
- The PL portfolio guides concern composition
- Conflicts identified during composition are analyzed and resolved
<table>
<thead>
<tr>
<th>Concern</th>
<th>Influence</th>
<th>Mapping</th>
</tr>
</thead>
<tbody>
<tr>
<td>Function Points</td>
<td>spec, design, impl</td>
<td>Component</td>
</tr>
<tr>
<td>NSN</td>
<td>spec, design, impl</td>
<td>Component</td>
</tr>
<tr>
<td>Glossaries and tables</td>
<td>spec, design, impl</td>
<td>Component</td>
</tr>
<tr>
<td>Development log</td>
<td>spec, design, impl</td>
<td>Component</td>
</tr>
<tr>
<td>Script management</td>
<td>spec, design, impl</td>
<td>Component</td>
</tr>
<tr>
<td>Technical reviews</td>
<td>spec, design, impl</td>
<td>Component</td>
</tr>
<tr>
<td>Test cases</td>
<td>spec, design, impl</td>
<td>Component</td>
</tr>
<tr>
<td>Defects</td>
<td>spec, design, impl</td>
<td>Component</td>
</tr>
<tr>
<td>Risk management</td>
<td>spec, design, impl</td>
<td>Component</td>
</tr>
<tr>
<td>MS-Office Compatibility</td>
<td>spec, design, impl</td>
<td>Component</td>
</tr>
<tr>
<td>Printing</td>
<td>spec, design, impl</td>
<td>Component</td>
</tr>
<tr>
<td>Project</td>
<td>spec, design, impl</td>
<td>Component</td>
</tr>
<tr>
<td>Portability</td>
<td>Arch, design, impl</td>
<td>Decision</td>
</tr>
<tr>
<td>Export</td>
<td>spec, design, impl</td>
<td>Aspect</td>
</tr>
<tr>
<td>Data dictionary</td>
<td>spec, design, impl</td>
<td>Component</td>
</tr>
</tbody>
</table>
Product Line Quality Model

- CAFÉ (from Concepts to Application in System-Family Engineering) deliverable
- Developed at the Fraunhofer IESE

- Elements:
  - Quality requirements: scenarios
  - Metrics
  - Architectural patterns
  - Use of means
PL-specific quality attributes

- Variability
- Derivability
- Reusability
- Generality
- Portability
- Security
- Rateability
- Integrability
- Correctness
- Evolvability
- Manageability
- Maintainability
- Architectural views and styles
  - PLQM and SEI’s *Views and Beyond*

<table>
<thead>
<tr>
<th>Quality attribute</th>
<th>View</th>
<th>Viewtype</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reusability, derivability, generality</td>
<td>Generalization, uses</td>
<td>Module</td>
</tr>
<tr>
<td>Modifiability</td>
<td>Decomposition, layers</td>
<td>Module</td>
</tr>
<tr>
<td>Evolvability</td>
<td>Decomposition, layers, Generalization</td>
<td>Module</td>
</tr>
<tr>
<td>Portability</td>
<td>Layers</td>
<td>Module</td>
</tr>
<tr>
<td>Security</td>
<td>Deployment</td>
<td>Allocation</td>
</tr>
</tbody>
</table>
Architecture modeling

- During the development of viewtypes, patterns, principles, techniques and tactics of the PLQM are incorporated
- CORE dimensions are considered (architectural decisions reinforce the PLQM)
- Commonality and variability is modeled
- Derivation of products specific architecture was done in C&C viewtype
- An aspectual view was included
PLA Evaluation

- Using the metrics and scenarios specified in the PLQM
- ATAM (Architecture Tradeoff Analysis Method)
- COSAAM (Concern-Oriented Software Architecture Analysis Method)
COSAAM (Concern-Oriented Software Architecture Analysis)

- Evaluation of concerns (crosscutting and non-crosscutting) in SA
- Measurement of scattering and tangling
- Highest tangling degree is 3 (of 52 concerns) for 3 modules
- Inherently crosscutting concerns had already been resolved as aspects or architectural decisions cohesive modules
- Accidentally crosscutting concerns are natural due to activities developed and artifacts generated during the domain analysis phase
Conclusions

- AOPLA encompasses a concern-oriented focus early in the process that allows for the identification, modeling of concerns (crosscutting and non-crosscutting) and their consideration into the PLA
Conclusions

- Obtention of a PLA that:
  - Fulfills the PL-specific quality attributes
  - Fulfills the products specified quality attributes
  - Allows the derivation of products
  - Supports commonality and variability
Questions?

karen@cimat.mx
k cortes@uv.mx

CIMAT, A.C.