A Methodological Approach to Domain Engineering for Software Variability Enhancement

Alexandre Bragança¹,² and Ricardo J. Machado³

¹ Dep. I&D, I2S Informática – Sistemas e Serviços SA, Porto, Portugal,
alexandre.braganca@i2s.pt

² Dep. Eng. Informática, ISEP, IPP, Porto, Portugal,
alex@dei.isep.ipp.pt
www.dei.isep.ipp.pt/~alex

³ Dep. Sistemas de Informação, Universidade do Minho, Guimarães, Portugal,
rmac@dsi.uminho.pt
http://www.dsi.uminho.pt/~rmac

Second Workshop on Method Engineering for Object-Oriented and Component-Based Development
Motivation

- Flexibility is a major quality of software systems
- Flexibility can be achieved by adopting variability realization techniques

*But*
- This also usually implies more complexity in the development process

- We address this problem with a domain engineering approach
Topics

1- Variability Realization Concepts and Techniques
2- Variability in the Software Development Process
3- Domain Engineering Approach to Software Variability
4- Case Study: Insurance Software System
5- Conclusion
In order to increase flexibility, variability realization techniques can be adopted.

<table>
<thead>
<tr>
<th>Pre-deployment</th>
<th>Post-deployment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre-processing of source code</td>
<td>Dynamic binding</td>
</tr>
<tr>
<td>Linking directives</td>
<td>Component infrastructure</td>
</tr>
<tr>
<td>Parameterization</td>
<td>Scripting languages</td>
</tr>
<tr>
<td>Inheritance</td>
<td>Reflection</td>
</tr>
<tr>
<td>Code Composition</td>
<td></td>
</tr>
</tbody>
</table>
Variability Concepts

- **Introduction Time (IT)** – The moment in the development process when the variability point is introduced
- **Binding Time (BT)** – The moment in the development process when the variability point is bound to a variant
- **Number of Variants (nV)** – The number of possible variants of the variability point
Supporting Variability: Case 1

- Example: Supporting more than one database in a system (non-functional concern)

This can be achieved by adopting pre-deployment techniques:

```c
#ifndef SQL_DB == SQL_DB_ODBC
...
#elif SQL_DB == SQL_DB_CLI400
...
#else
#error ...
```
Variability Measure: Case 1

(Simple) Measure of Variability (Flexibility): $IT \times BT \times nV = 3 \times 5 \times 2 = 30$
Variability Evaluation: Case 1

- In this case, the application is more flexible but:
  - The code that varies is scattered across the application;
  - Complexity is high;
  - The concern is not managed as a unity;
  - Manageability is low;
  - It’s difficult, if not impossible, to predict the impact of change and evolution.
One alternative is to promote all the database functionality into a component or subsystem.

```
| «interface» |
| SQL2_Service |
| +SQL2_Init() |
| +SQL2_Terminate() |
| +SQL2_Connect() |
| +SQL2_BeginTrans() |
| +SQL2_Commit() |
| +SQL2_RollBack() |
| +SQL2_Exec() |
```
Variability Measure: Case 2

1 - Requirements

2 - Analysis

IT

3 - Design

4 - Implementation

5 - Compilation

Pre-deployment

6 - Linking

Post-deployment

7 - Installation

BT

8 - Execution

(Simple) Measure of Variability (Flexibility): IT * BT * nV = 3 * 8 * 2 = 48
Variability Evaluation: Case 2

- The non-functional concern has been promoted to a component with a well known interface.
- Variability is handled within the component or subsystem.
- The degree of variability is higher (later variability times).
- This non-functional concern becomes more manageable and less complex.
- The aim of this component/subsystem is not to be reused but to support variability within an application in a manageable way.
1- Variability Realization Concepts and Techniques

2- Variability in the Software Development Process

3- Domain Engineering Approach to Software Variability

4- Case Study: Insurance Software System

5- Conclusion
Variability vs. Development Process

- Product-line (PL) approaches deal with Commonality and Variability. Regarding reuse and variability, software can be classified as:

<table>
<thead>
<tr>
<th>Decomposition</th>
<th>Composition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single Product</td>
<td>Component market</td>
</tr>
<tr>
<td>Application System Variants</td>
<td></td>
</tr>
<tr>
<td>Application System Suite</td>
<td></td>
</tr>
<tr>
<td>Fairly Independent Application Systems</td>
<td></td>
</tr>
<tr>
<td>Product Population</td>
<td></td>
</tr>
</tbody>
</table>

Method Engineering Workshop
Product lines are based on a two level engineering process.

The Domain Engineering (DE) Level applies Methods to support the development of artefacts to be reuse (and their variability).

The Application Engineering Level reuses artefacts from the Domain level to compose new applications.

This is a compositional approach to development.
PL Two Level Engineering Process

Domain Engineering | Application Engineering

- Requirements
- Analysis
- Design
- Implementation

<table>
<thead>
<tr>
<th>Pre-deployment</th>
<th>Post-deployment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compilation</td>
<td>Installation</td>
</tr>
<tr>
<td>Linking</td>
<td>Execution</td>
</tr>
</tbody>
</table>
PL High-Level Engineering Process

Generalization Process
(Commonality and Variability)

Specialization Process
(Selecting and Completing Features)

Customization Process

Domain Engineer

Domain Engineering

Domain Artefacts
(components DSL etc.)

Application Engineering

Application A

Application B

Application Z

Customer

Application Domain Specialist

Application Deployment

Running Application A

Software-house
Domain Engineering (DE) encompasses a proactive method that:

- Analysis the domain of the product line with the goal of developing reusable artefacts;
- Develops reusable artefacts anticipating the needs of future applications;
- Incorporates variability into the design of artefacts in order to support the adaptability needs of future applications.
1- Variability Realization Concepts and Techniques
2- Variability in the Software Development Process
3- Domain Engineering Approach to Software Variability
4- Case Study: Insurance Software System
5- Conclusion
Given the fact that flexibility is achieved through variability realization techniques and that domain engineering has proven to be a satisfactory method to manage variability.

- We propose a new approach to application design by adopting domain engineering methods within the development process in order to manage variability points and thus achieve flexibility at the application and development levels.
The Approach Process

Application Engineering | Domain Engineering
---|---
Requirements | Analysis
Analysis | Design
Design | Implementation
Implementation | Compilation
Compilation | Linking
Pre-deployment | Post-deployment
Installation | Execution
From the application Requirements and Analysis phases should result the identification of variability concerns that should be engineered at the domain level.

The domain engineering level implements the variability concerns that are integrated into the application at the implementation phase.
The Approach Process Outline

Domain Engineering

- Analysis
- Initial Design
- Design
- Implementation

Domain Scope and Architecture

Variability Concern Interface

Application Engineering

- Requirements
- Analysis
- High Level Design
- Detailed Design
- Implementation

A Methodological Approach to Domain Engineering for Software Variability Enhancement
Topics

1- Variability Realization Concepts and Techniques
2- Variability in the Software Development Process
3- Domain Engineering Approach to Software Variability
4- Case Study: Insurance Software System
5- Conclusion
Case Study: Insurance System

- Register Insured Persons
  - «extends»
  - «extends»
- Register Insured Entities
  - «extends»
- Register Policyholder
  - «uses»
  - «uses»
  - «uses»
- Select Captive Agent
  - «extends»
  - «extends»
  - «extends»
- Register Policy
  - «extends»
  - «extends»
  - «extends»
  - «extends»
- Select Health Coverage
  - «extends»
  - «extends»
- Select Automotive Coverage
  - «extends»
  - «extends»
- Select Life Coverage
  - «extends»
- Register Beneficiaries
  - «extends»
- Register Drivers
  - «extends»
- Issue Deposit Premium

A Methodological Approach to Domain Engineering for Software Variability Enhancement
Insurance domain requires rapidly changes to business products

These changes imply functional changes to the system

Non computer-specialists at the customer site should be able to make these changes

This is a case for end-user DSLs
Software Development Process

Specialization/Generalization Process

Generalization Process (Commonality and Variability)

Specialization and Customization Process

Application System with Open Variants

Application Domain Specialist

Deployment

Running Application (Several Business Products)

Customer

Software-house

Domain Engineer

Domain Engineering

Application Engineering

Application/Domain Knowledge

Application Engineer
• The variability in this kind of systems essentially reflects the need to externalize business rules from the system.
Topics

1- Variability Realization Concepts and Techniques
2- Variability in the Software Development Process
3- Domain Engineering Approach to Software Variability
4- Case Study: Insurance Software System
5- Conclusion
Conclusion

- We propose a method based on domain engineering to enhance flexibility in software systems.
- This is an innovating approach to domain engineering methods adoption.
- This method applies to highly configurable and adaptable systems of specific domains.
Future Work: Method

• **Process:**
  – Detail and validate the presented process
    • *Adopt SPEM notation to describe meta-model*

• **Technology:**
  – What techniques and technologies to adopt? - DSLs, generative approaches, design patterns, reflection…
    • *Selection should be based on the analysis of the variability concern*

• **Model:**
  – Adopt UML 2.0 as central notation
    • *Is this a case for DSLs (vs. UML)?*
Thank you.

Alexandre Bragança\textsuperscript{1,2} and Ricardo J. Machado\textsuperscript{3}

\textsuperscript{1} Dep. I&D, I2S Informática – Sistemas e Serviços SA, Porto, Portugal, \newline
alexandre.braganca@i2s.pt

\textsuperscript{2} Dep. Eng. Informática, ISEP, IPP, Porto, Portugal, \newline
alex@dei.isep.ipp.pt \newline
www.dei.isep.ipp.pt/~alex

\textsuperscript{3} Dep. Sistemas de Informação, Universidade do Minho, Guimarães, Portugal, \newline
rmac@dsi.uminho.pt \newline
http://www.dsi.uminho.pt/~rmac

Second Workshop on Method Engineering for Object-Oriented and Component-Based Development