An Analysis of Variability Modeling and Management Tools for Product Line Development

Rafael Capilla, Alejandro Sánchez
Universidad Rey Juan Carlos de Madrid

Juan C. Dueñas
Universidad Politécnica de Madrid

SPAIN
Motivation

- Time to market of complex software systems and Product Family/Product Line approaches demand variability-based solutions.

- **Software Variability** is key for PL development.

- The increasing number of variation points needs of techniques and tools able to manage the variability of systems.

- Managing variability at runtime becomes a complex problem that has to be addressed.

- Different variability techniques and approaches have been proposed and used but a lack of agreements is missing.

- Several tools for the same goal.

R. Capilla, A. Sánchez, J.C. Dueñas, SVM-WS’07, Helsinki, Finland
FODA models (1990) are used for modeling and representing the common and variable parts of software systems.

- Describe the external and visible properties of systems
- Requirements are modeled as “features”
- Features are used to describe the variability and the relationships among these features
- Complex dependencies are not straightforward

R. Capilla, A. Sánchez, J.C. Dueñas, SVM-WS’07, Helsinki, Finland
Variability concepts

- Variation point: An area of a software system affected by variability
- Variant: The alternatives defined for each particular VP
- Variability in space (allowed set of product configurations) and variability in time (when the variability is realized)
- Extensibility. Closed VP (variants are known at pre-deployment time) Open VP (VP and variants can be added at runtime)
- Variability realization: Different techniques proposed for implementing the variability model (i.e.: inheritance, aggregation, parameterization, etc...)

R. Capilla, A. Sánchez, J.C. Dueñas, SVM-WS’07, Helsinki, Finland
Dependencies and constraints limit the number and type of configurable products from the number of possible configurations.

Dependency links introduce a factor of complexity that variability models have to manage.

Several dimensions and categories are possible in the variability model.

Complex and simple dependencies are possible.

Dependencies can be used to establish traceability issues.

R. Capilla, A. Sánchez, J.C. Dueñas, SVM-WS'07, Helsinki, Finland
A feature may depend on other features

- AND, OR, XOR boolean relationships
- Requires and excludes dependencies
  
  ![Diagram showing dependencies between features F1 and F2: Needs, Not in the same product, Depends, Modifies, Behavior of F2]

- Usage, Modification, Others [LEE & KANG, ICSR2004]

R. Capilla, A. Sánchez, J.C. Dueñas, SVM-WS’07, Helsinki, Finland
## Dependencies in Linux systems

<table>
<thead>
<tr>
<th>Release</th>
<th>Codename</th>
<th>Date</th>
<th>Num. of packages</th>
<th>Distinct versions</th>
<th>Dependencies</th>
<th>RP</th>
<th>RD</th>
<th>RP-RD</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.3</td>
<td>Bo</td>
<td>1997-06-02</td>
<td>2088</td>
<td>1402</td>
<td>5111</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>2.0</td>
<td>Hamm</td>
<td>1998-07-24</td>
<td>2757</td>
<td>1946</td>
<td>7379</td>
<td>1.32</td>
<td>1.44</td>
<td>0.12</td>
</tr>
<tr>
<td>2.1</td>
<td>Slin</td>
<td>1999-03-09</td>
<td>3601</td>
<td>2691</td>
<td>10841</td>
<td>1.31</td>
<td>1.47</td>
<td>0.16</td>
</tr>
<tr>
<td>2.2</td>
<td>Potato</td>
<td>2000-08-14</td>
<td>5583</td>
<td>4311</td>
<td>19284</td>
<td>1.55</td>
<td>1.78</td>
<td>0.23</td>
</tr>
<tr>
<td>3.0</td>
<td>Woody</td>
<td>2002-07-19</td>
<td>10771</td>
<td>8693</td>
<td>44164</td>
<td>1.93</td>
<td>2.29</td>
<td>0.36</td>
</tr>
<tr>
<td>3.1</td>
<td>Sarge</td>
<td>2005-06-05</td>
<td>19300</td>
<td>15305</td>
<td>96981</td>
<td>1.79</td>
<td>2.20</td>
<td>0.40</td>
</tr>
</tbody>
</table>

![Graph 1](image1.png)

![Graph 2](image2.png)

R. Capilla, A. Sánchez, J.C. Dueñas, SVM-WS’07, Helsinki, Finland
Variability management

- Deals with the processes to create, use, modify, maintain and document the variability model across different stages of the lifecycle.

- Variation points and their dependencies have to be maintained.

- Some approaches / tools provide adequate visualization support.

- Integration with source code is needed for product derivation.

- Evolution of products and VPs must be supported.

- Feature categorization for PL development facilitates the visualization of feature models.

R. Capilla, A. Sánchez, J.C. Dueñas, SVM-WS'07, Helsinki, Finland
The scope of the study carried out was focused on variability and management and modeling tools.

The analysis describes the main characteristics of the tools from the information gathered from the authors (interviews, discussions).

No tool evaluation was done because two of them are not available for the general public.

No cross-comparison and ranking between tools was made.
GEARS is a commercial SPL development tool from BigLever Inc.

GEARS enables modeling optional and varying features for producing different products.

GEARS distinguishes between features at the domain modeling level and variations points at the implementation level.

Components with variation points are turned into reusable assets that are automatically composed and configured into product instances.

Dependencies are expressed as relational assertions which may contain 3 or more features and relationships.

Specific product profiles to select the choices for each product in the feature model.

R. Capilla, A. Sánchez, J.C. Dueñas, SVM-WS’07, Helsinki, Finland
Tools for modeling and managing variability in PL development GEARS

Feature models
A product feature profile is created for each product in the portfolio.

Configurable assets and variation points
Gears provides a language for programming each VP.

A product configurator automatically produces the products in the portfolio composing the assets and configuring the VP.

Gears manufactures each product based on specific profiles and reflect the changes when assets are modified.

R. Capilla, A. Sánchez, J.C. Dueñas, SVM-WS’07, Helsinki, Finland
V-Manage tool for small/medium organizations that want to implement a PL and supports SFE in the context of MDA

V-manage consists of 3 modules:

- **V-define**: Represents the variation model (i.e.: decision model) and its relationships
- **V-Resolve**: Builds application models and sets the values of the decision model to produce a suitable configuration
- **V-Implement**: Produces the reusable components
Tools for modeling and managing variability in PL development

Variation Model

User
V-DEFINE

Variability is resolved

Application Model

PL Model

Concrete system Model

R. Capilla, A. Sánchez, J.C. Dueñas, SVM-WS’07, Helsinki, Finland
Tools for modeling and managing variability in PL development

V-Manage

File

Element

Variants

Attributes of the decision

R. Capilla, A. Sánchez, J.C. Dueñas, SVM-WS’07, Helsinki, Finland
Tools for modeling and managing variability in PL development

V-Manage

Cond. associated to parameters

Source code

R. Capilla, A. Sánchez, J.C. Dueñas, SVM-WS'07, Helsinki, Finland
COVAMOF (ConIPF Variability Modeling Framework) is a tool for representing VP and variants at all levels of abstraction.

- Supports dependencies and a hierarchical variability model. Complex dependencies are defined as dynamically analizable dependencies.

- 5 types of VP are supported.

- COVAMOF Variability View (CVV) represents the view of the variability for PF artifacts and unifies this variability on all layers. CVV also models the dependencies to restrict the binding of VPs.

- The Mocca tool supports multiple variability views in COVAMOF.

R. Capilla, A. Sánchez, J.C. Dueñas, SVM-WS’07, Helsinki, Finland
Tools for modeling and managing variability in PL development

COVAMOF

Legend
- Variation Point
- Variant
- Realization
- Dependency
- Dependency Interaction

Product class
- Full
- Custom
- Basic

Feature Layer
- Notepad Interface
- Word Interface
- Built-in Editor
- Mail Manager
- Network Interface
- Database Interface
- File Storage

Architecture Layer
- Editor Component
  - Notepad
  - Built-in
  - Word
- Processor Usage
- Performance
- Memory Consumption
- Persistency Component
  - Database
  - Files

R. Capilla, A. Sánchez, J.C. Dueñas, SVM-WS’07, Helsinki, Finland
Tools for modeling and managing variability in PL development

R. Capilla, A. Sánchez, J.C. Dueñas, SVM-WS’07, Helsinki, Finland
The variability Modeling Web Tool is a Web-based tool with PHP+AJAX for modeling variability for product line development.

- FODA trees for visualizing the variability model
- Dependencies supported: Boolean connectors and requires, excludes
- Dependency checking before producing the product configuration
- Computes the number of allowed products
- Automatic documentation as PDF documents
- VP and variants are included in the code assets

R. Capilla, A. Sánchez, J.C. Dueñas, SVM-WS’07, Helsinki, Finland
Tools for modeling and managing variability in PL development

R. Capilla, A. Sánchez, J.C. Dueñas, SVM-WS’07, Helsinki, Finland
The AHEAD (Algebraic Hierarchical Equations for Application Development) tool suite supports the development of PL by means of compositional programming and based on the GenVoca methodology for incrementally add features to product family members.

The key tool in AHEAD tool suite is the composer, which expands AHEAD equations to yield the target system.

AHEAD distinguishes between “product features” and “built-in features”.

AHEAD uses a step-wise refinement process. Refinements are packaged in layers. The base layer contains base artifacts which are enhanced with specific features.

Automatic derivation process.

R. Capilla, A. Sánchez, J.C. Dueñas, SVM-WS’07, Helsinki, Finland
Tools for modeling and managing variability in PL development AHEAD

Features are directories!!

R. Capilla, A. Sánchez, J.C. Dueñas, SVM-WS’07, Helsinki, Finland
Composing Features

> composer --equation=baseRef
#baseref.equation
base
ref

\[
\text{basemuladd} = \text{add} \cdot \text{mul} \cdot \text{base}
\]

> composer -target=basemuladd add mul base
Several successful examples have demonstrated how Product Lines are useful for industry to meet market demands.

Examples of savings and reuse producing multiple products in parallel are provided in the literature:

- Around the 80% of the code produced by Engenio (a firm to high performance disk storage systems) is common to 82 products of the firm.

Tools are becoming key pieces for managing the increasing amount of variability in PFE and for automating the derivation process.

R. Capilla, A. Sánchez, J.C. Dueñas, SVM-WS’07, Helsinki, Finland
Conclusions

- Most of the tools examined share many similarities.
- Lack of a unified variability approach that leads to several tools/approaches, and notations.
- Visualizing hundreds of VP is a limitation to overcome.
- Need to handle complex dependencies.
- Incompatible versions of products must be checked before product derivation.
- Managing variability at runtime is hard.

R. Capilla, A. Sánchez, J.C. Dueñas, SVM-WS'07, Helsinki, Finland
Future needs – Discussion topics

- Integration with other SE development tools and SCM tools
- More visualization capabilities
- Support for runtime variability
- Agreement on standard notation?
- Estimation of the cost of products from the product portfolio
- Better integration from modeling and configuration to product synthesis
- Identify and extract features from code (reverse engineering, feature location) to integrate them with an existing feature model

R. Capilla, A. Sánchez, J.C. Dueñas, SVM-WS’07, Helsinki, Finland