**VARiability In safety-critical Embedded Systems**

ARTEMIS-2011-1
Project Number: 295397

Variability Modeling and its Standardization

by Øystein Haugen, SINTEF
MDD4DRES @ Aber Wrac'h 2. September 2014
Overview of the talk

- Short bio of Øystein Haugen
- What is a Product Line and how should its variability be described?
- About CVL (Common Variability Language) its history and standardization effort.
- Applying variability modeling to the testing of product lines
- Understanding variability models – how easy is it?
- Some thoughts about the future of variability modeling.
University and Research Inst.
• 80-81: UiO, Research assistant for Kristen Nygaard
• 81-84: Norwegian Computing Center
  – Simula-machine
• 97: Practitioners’ verification of SDL systems (Dr. scient.)
• 98-03: Ifi, UiO as Part time Associate Professor
• 04-07: Associate Professor at Ifi (100%)
• 07-: Senior Researcher SINTEF
  – Projects on modeling languages e.g. for variability, train control and pay rolls
• 07-: Assoc. Professor at Ifi (20%)
• 10: General Chair MODELS 2010
• 12-15: VARIES (Variability Modeling)

Industry and Standardization
• 84-88: SimTech, typographical applications
• 88-90: ABB Technology
  – SDL, prototype SDL tool, ATC
• 91-97: Independent Consultant
• 96-00: Rapporteur ITU for MSC
• 97-03: Ericsson, NorARC
• 99-11: OMG wrt. UML 2.0
  – Responsible for UML 2.x chapter on Interactions
• 09 - 13: OMG CVL – Common Variability Language
  – Coordinated joint submission team
Agenda

• Short bio of Øystein Haugen
• What is a Product Line and how should its variability be described?
• About CVL (Common Variability Language) its history and standardization effort.
• Applying variability modeling to the testing of product lines
• Understanding variability models – how easy is it?
• Some thoughts about the future of variability modeling.
What do we mean by “variability”?

- **Product Line variance**
  - often variants of the same software base

- **Cross-cutting variability**
  - often variability is orthogonal to the software design
  - variability needs are discovered after the first software design

- **The variability designer is not always the software designer**
  - division of labor and of competences
# Traditional ways to model variability

<table>
<thead>
<tr>
<th>Constructs</th>
<th>Framework/Configuration</th>
<th>Union-of-all-systems</th>
<th>Domain Specific Languages</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>How?</strong></td>
<td>By mechanisms of a general language</td>
<td>As annotations to a language</td>
<td>By the specific language mechanisms</td>
</tr>
<tr>
<td><strong>Functions</strong></td>
<td>Function, Type, Inheritance, Template, Plugin</td>
<td>Pragma, Stereotype</td>
<td>Proprietary language constructs</td>
</tr>
<tr>
<td>unforeseen modeling needs</td>
<td>Just enhance the final model</td>
<td>Enhance the product line model</td>
<td>If not expressible, enhance the language</td>
</tr>
</tbody>
</table>

**Constructs**
- Function
- Type
- Inheritance
- Template
- Plugin
- Pragma
- Stereotype
- Proprietary language constructs

**How?**
- By mechanisms of a general language
- As annotations to a language
- By the specific language mechanisms

**Unforeseen modeling needs**
- Just enhance the final model
- Enhance the product line model
- If not expressible, enhance the language
Agenda

• Short bio of Øystein Haugen
• What is a Product Line and how should its variability be described?
• About CVL (Common Variability Language) its history and standardization effort.
• Applying variability modeling to the testing of product lines
• Understanding variability models – how easy is it?
• Some thoughts about the future of variability modeling.
CVL - Common Variability Language

Formal Analysis

Base: DSL e.g. TCL

CVL Model

unify

validate

apply
Common Variability Language (CVL)

- **Generic & Standardized**: CVL
- **Focused on a domain**: DSL

Variability model

- Specification in CVL of base model variabilities
- Selection of a set of choices in the variation model

Base model

- Product line model in any MOF-compliant language

Resolved models

- Product models fully described in the base language.
- All regular base language tools can be applied to these models

Execute CVL
CVL Architecture

- Configurable Units
- Variability Realization
- Variation Points
- VSpecs
- Constraints
- Resolutions
- Base Model
- Variability Interfaces
- Variability Abstraction
Variation Points over base model

CVL variation points

SYSML (base model) elements

- Variability in this example:
  - Part EmergencySupply is optional
  - Part HighSpeedConnector is optional
  - Port EmgPowerCtrl on block Printer is optional
  - Value of attribute threshold in block EmergencyPower is variable
Variation points in CVL

- Variation Points refer to Base objects
- Variation Points define the base model modifications precisely
- There are different kinds of Variation Points
  - Existence
  - Value assignment
  - Substitution
  - Opaque variation point
  - Configurable Unit
VSpec trees and binding

VSpec tree
- choice
- variable
- constraint

Variation points

MainSupply:MainPower
- Attributes
- Operations

EmgSupply:EmgPower
- Attributes
- Operations

inputSection
- Attributes
- Operations

HighSpeedConnecter
- Attributes
- Operations

threshold:Int
- Attributes
- Operations

Printer
- Attributes
- Operations

MainPower
- Attributes
- Operations

EmgPower
- Attributes
- Operations

MainPower «Block»

Values
- powerCtrl

EmgPower «Block»

Values
- threshold:int
- powerCtrl

HighSpeed & threshold>100 ➔ EmergencyPower

VSpec trees and binding

Printer «Block»

Values
- powerCtrl

EmergencyPower «Block»

Values
- threshold:Int
- powerCtrl

Variation points

- ObjectExistence
- ObjectExistence
- SlotValueAssignment
- ObjectExistence

Attributes
- threshold
- powerCtrl

Operations
- MainSupply:MainPower
- EmgSupply:EmgPower

InputSection

- highSpeedConnector

Operations
- MainSupply:MainPower
- EmgSupply:EmgPower

Operations
- MainSupply:MainPower
- EmgSupply:EmgPower
VSpecs in CVL

- VSpecs (Variation Specifications) describe the abstract variability
- Every Variation Point is bound to exactly one VSpec
- VSpecs come in different kinds:
  - Choice
  - Variable
  - VClassifier
  - CVSpect
Constraints in CVL

- CVL includes BCL – Basic Constraint Language – for expressing constraints on the VSpec tree
  - Propositional logic is supported
- CVL also has the opportunity to let you apply other constraint languages like OCL
Resolution

Resolution model:
- False
- True
- 90

Variation points:
- HighSpeed & threshold>100 ➔ EmergencyPower

Model:
- Printer
- MainSupply:MainPower
- EmgSupply:EmgPower
- HighSpeed
- EmergencyPower
- threshold:Int
- :ObjectExistence
- :ObjectExistence
- :SlotValueAssignment
- :ObjectExistence

Operations:
- powerCtrl

MainPower and EmgPower:
- Values

InputSection:
- highSpeedConnector

mainSupply:MainPower and emgSupply:EmgPower:
- Operations
Variability Resolution in CVL

- VSpecResolution elements refer to VSpecs
- The set of valid Resolutions is restricted by the constraints
- Represent information necessary to materialize product models
  - Actual yes/no decisions on Choices
  - Actual values to Variables
  - Instances of VClassifiers
  - Configurations of CVSpecs/Configurable Units
Materialization

HighSpeed & threshold>100 → EmergencyPower

Resolution model

False
True
90

Variation points

Materialization

threshold=90
CVL – more information

- Please look at [http://variabilitymodeling.org](http://variabilitymodeling.org)
- There you will find e.g.
  - The Revised Submission of CVL
  - Tutorial slides from SPLC tutorial on CVL
  - Links to supplementary, and historical material
Standardizing

How standards proliferate:
(See: A/C chargers, character encodings, instant messaging, etc.)

**Situation:**
There are 14 competing standards.

14?! Ridiculous!
We need to develop
one universal standard
that covers everyone’s
use cases.

Yeah!

Soon:

**Situation:**
There are 15 competing standards.
Why standardize?
Why standardize a language?

• Common terms and interpretation
  – across persons, teams, companies and cultures
    • Experience SISU project
      – Very large SDL specification ported from Alcatel to Kongsberg
    • Experience MSC
      – We have a Korean translation of MSC 2000
  – across computers (portability)
    • Experience Simula
      – We ported the exact same code on at least 5 machines without changing a single line of code around 1980

• Common teaching material
• Common libraries
• Common and open reviewing process
CVL in OMG

- Late 2009, Request for Proposals (RFP)
- September 2012 Revised submission
- Late 2010, Initial Submission
- Spring 2013, Finalization?

JOINT SUBMISSION TEAM
initiators from the MoSiS project

tool vendors
- atego
- IBM
- pure-systems

users & consultants
- SINTEF
- tecnalia
- SINO
- TATA
- TATA CONSULTANCY SERVICES
- THALES

research institutes and universities
- University of Waterloo
- Inria
- CEA
- List
- IT UNIVERSITY OF COPENHAGEN
- Fraunhofer FOKUS
- ARTEMIS

SINTEF
In Variability Modeling there has been
  – focus on notation
  – focus on the abstraction layer (feature models)
  – few attempts to standardize formally

but for some reason now things start to happen
  – Separate language (variability language as such)
    • CVL (our baby)
  – Amalgamated approaches (variability combined into another language)
    • AUTOSAR
    • Matlab Simulink
    • SysML
Success is not guaranteed

- Risk 0: Nobody cares about your proposed standard
  - This is why I prepared OMG over many years (since 2005)
- Risk 1: You meet strong opposition
  - This happened when we wanted OO into SDL (1989-1992)
  - This happened again when we wanted UML more precise (1999)
- Risk 2: You run out of time and money
  - Research projects are seldom more than 4 years
  - Standardization may easily take longer
- Risk 3: You achieve a standard, but nobody makes tools
  - This has happened to many standards
- Risk 4: There is a standard, there are tools, but no users
  - May be your compromises were just not good enough ....
Challenges (1)

- For the academic:
  - Cannot invent new things all the time
  - Needs to have the totality in mind
    - not only a very limited part of the domain/language
    - no assumptions of simplification
  - Little personal credit for authoring a standard
- For the industrial researcher:
  - The tool prototype is lagging behind the language
    - CVL 1 Tool is supporting something similar to OMG CVL but ... 
    - ATL and MOFScript were not quite supporting the OMG standard
  - The project terminates with no standard in sight
• For the tool vendor:
  – The compromises of the standardization increases the need to modify the tool they already have
  – They always want the simplest solution that they believe their customers are asking for
    • The solution perceived as simplest may also be the most special

• For the industrial early adopter:
  – Tool support is lacking!
    • As an Ericsson representative I experienced that with SDL, MSC and UML
    • The more we are asking for of the standard, the more the tooling is lagging
BigLever (Krueger) has one US patent and a couple of patent applications

Krueger claims that any tool that supports CVL may infringe upon these patents

Krueger does not want to be more specific than this and says that the information is in the patent papers

My assessment is that CVL does not infringe the BigLever patent
The BigLever patents

  - **Abstract:** A system and method for the mass customization of software includes a software production line infrastructure, development environment, and actuator. The infrastructure organizes the software production line using a feature declarations component, a product definitions component, and automatons component and a partition composition component. The development environment browses, displays, organizes, edits, and maintains the infrastructure. The actuator actuates the software production line to produce custom software products.

- **US Patent Application, 12/273,352, Filed Nov 18, 2008.** This contains claims that are still pending from the original patent listed above.

What happens now with CVL?

Common Variability Language (CVL)

OMG Revised Submission

OMG document: ad/2012-08-05

**Submitters**
- IBM
- Fraunhofer FOKUS
- Thales
- Tata Consultancy Services

**Supporters**
- SINTEF
- University of Oslo
- Tecnalia Research & Innovation
- University of Waterloo
- IT University of Copenhagen
- INRIA
- CEA
- Atego
- Pure-systems

Primary Contact:
Øystein Haugen, SINTEF
oystein.haugen@sintef.no

Date: August 13, 2012
CVL in the OMG process

- CVL Revised Submission exists
- CVL was presented to the OMG Architectural Board in December 2012
  - The Architectural Board of OMG recommended CVL technically
- The OMG Business Committee decides on the business aspects of the standard.
  - They also considers the patent infringement possibilities
- The OMG Business Committee has not reached its final verdict and is not likely to do so within reasonable time
The Future of CVL

- Common or Chaos?
- Variability Language or a Variety of languages?
- Leveraging the community or succumbing to Limitations?
• Short bio of Øystein Haugen
• What is a Product Line and how should its variability be described?
• About CVL (Common Variability Language) its history and standardization effort.
• Applying variability modeling to the testing of product lines
• Understanding variability models – how easy is it?
• Some thoughts about the future of variability modeling.
Testing Product Lines

- The following slides are based entirely on work done by Martin Fagereng Johansen
- Documentation in his PhD Thesis:
  - Testing Product Lines of Industrial Size: Advancements in Combinatorial Interaction Testing
  - [https://www.duo.uio.no/handle/10852/38097](https://www.duo.uio.no/handle/10852/38097)
- Challenges
  - Product Lines may have many features
    - The number of possible products may be very large!
  - With physical systems changing configuration is time consuming
  - With limited resources (in particular: time) what would be the best configurations to test in order to say something sensible about the whole product line?
• It is normally not sufficient to test each feature (property) in isolation
  – sometimes even knowing how to test them in isolation is difficult
• Often errors occur in the combination of several features
  – CIT establishes a coverage array such that every t-tuple of features are covered
  – This works well when errors show up when the problematic combination (t-tuple of features) is present
    • Every combination of the t-tuple must be covered, not only when the features are present, also where they are not present
• To find a small number of configurations that cover all t-tuples of all features for a product line, is in general intractable, but in practice it is more often than expected, feasible
## A Coverage Array

<table>
<thead>
<tr>
<th>Feature\Product</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
<th>21</th>
<th>22</th>
</tr>
</thead>
<tbody>
<tr>
<td>EclipseSPL</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>WindowingSystem</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Win32</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>GTK</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Motif</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Carbon</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Cocoa</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>OS</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>OS_Win32</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Linux</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>MacOSX</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Hardware</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>x86</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>x86_64</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Team</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>CVS</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>CVS_Over_SSH</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>CVS_Over_SSH2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>SVN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Subversive</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Subclipse</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Subclipse_1.4_x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Subclipse_1.6_x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>GIT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>EclipseFileSystem</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Local</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Zip</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>
The Empirics: Why is CIT useful?

Kuhn et al. 2004
ICPL: The Open-Source Tool

- The tool is available as open-source
- It is very fast compared to other available tools
  - but finding coverage arrays still take considerable real time
- It is considered the reference tool for creating coverage arrays
  - some improvements have been suggested by other scientists
- Found problematic combinations in Eclipse configurations that Eclipse testing had not found
- Revealed problems in our own product line model in CESAR
  - problems that would have been difficult to spot without it
- Industrial experiments made us enhance the technique by introducing weights on variants
The CESAR/ABB Model

- Composite structure(s)
- «SUT» (Safety Module)
- «Test Component» (Motor Controller)
- «Test Component» (Motor)
- State Machine(s)
The CESAR example architecture

Variability and domain design

- Combinatorial Coverage arrays
- CVL Resolution
- The Product Line: CVL Abstraction Layer

Derivation

- The Product model with tests
- Generate

Deployment

- Product Simulation Code
- Generate
- Compile and run
- Execute tests
- Executable JUnit Testing Code
• Short bio of Øystein Haugen
• What is a Product Line and how should its variability be described?
• About CVL (Common Variability Language) its history and standardization effort.
• Applying variability modeling to the testing of product lines
• Understanding variability models – how easy is it?
• Some thoughts about the future of variability modeling.
A Skoda Yeti car can have the following combination of features: Heated-Front-Pane and Sunset and Panorama-Roof and Styling-Package?
Can you answer the questions?

A Skoda Yeti car can have the following combination of features:
Heated-Front-Pane and Sunset and Panorama-Roof and Styling-Package?

\[
\text{not (Parking Heater and Heated Front Pane)}
\]
\[
\text{not (Panorama Roof and Offroad Styling)}
\]
Findings to be published at MODELS

- This work was done in cooperation with Iris Reinhartz-Berger and Kathrin Figl
- We used our student classes for an online web survey where comprehension was one focus
- We distinguished between "pros" and "amateurs" (familiar or non-familiar with feature modeling)
- We also distinguished between "tree-structured" and "constraint-oriented"
Pros and amateurs understood basic mechanisms such as mandatory, optional and alternatives equally well
  – but amateurs expressed more uncertainty
Constraints were significantly better understood by pros
Pros spent more (!) time than amateurs
  – This felt counterintuitive, but one interpretation may be that pros sometimes recognized the complexity of the question and hesitated to answer immediately
Agenda

• Short bio of Øystein Haugen
• What is a Product Line and how should its variability be described?
• About CVL (Common Variability Language) its history and standardization effort.
• Applying variability modeling to the testing of product lines
• Understanding variability models – how easy is it?
• Some thoughts about the future of variability modeling.
After realizing that the CVL standardization would be stalled for an indefinite duration of time, we decided to move forwards without always adhering to CVL.

We called the resulting improvements BVR (Base Variability Resolution).

Our experience base was a cooperation with fire detection and security company Autronica.

The following results will be published in SAM 2014 and was done together with Ommund Øgaard (Autronica).
The Autronica Product Family

**AutroMaster**
Top level graphical presentation system running on Linux PCs

**FireAlarmSystem**
System consisting of embedded fire panels with logics and LCD menu system.

**AutroSafe**
High end networked Fire Alarm System for large demanding applications

**Autroprime**
Medium range standalone Fire Alarm System

**FieldEquipment**
Smoke detectors, manual call points, sounders, IO units etc.
The Market Domains of Autronica

Autronica – turnover 2010

- Turnover 2010: NOK 760 mill

- The maritime market: 27.6%
- The petrochemical, oil & gas market: 33.6%
- Onshore market – Norway: 28.8%
- The international onshore market: 10.1%
Autronica has two main general products: AutroSafe and AutroPrime. Each of these products can be seen as a separate product line. Configuring an AutroXXX installation is in itself a challenge:
- Many items to be configured
- Not only choices but also connections
- Long distance (Caribbean, etc)
Testing, analyzing and certification are important aspects.
Autronica are upgrading their general products and will bring in successful technology from VARIES.
Issue 1: The Target

Duplicating choices – not legal CVL

Unique choices, supplemented by constraints
Constraints are about targets

- Constraints are about targets and not about decisions
- VSpecRefs should always talk about the targets
- We want that duplicated names on VSpecs indicate that they are decisions on the same target
Issue 2: VSpec Type

Note that hp140 refers to any hp140 instance.
How to describe narrowing (2)?

- Narrowing can be described by a constraint on the instance
- This is compact and flexible
- It does move the style from tree-style to constraint-style

narrowing by constraint
Issue 3: Realization (additive, nested)
Problem is that these fragment substitutions do not know the resolution of the Engine!
Realization (staged realization)
No immediate standardization of the separate (CVL) approach
Possibly we will see amalgamation of variability into existing languages
  – SysML
  – AUTOSAR
  – Matlab
Researchers and practitioners will continue to make their own solutions