Capturing Variability in Space and Time with Hyper Feature Models

Christoph Seidl
Ina Schaefer
Uwe Aßmann
Background
- Software Ecosystems (SECOs) and Variability
- Formalization of Feature Models

Problem: No Variability in Time in Feature Models

Contributions
- Hyper Feature Models (HFM}s)
- Version-Aware Constraint Language
- Automatic Version Selection

Conclusion: Capturing Variability in Space and Time with HFM}s
Background

- Software Ecosystems (SECOs)
  - Family of closely related software systems
  - E.g., Eclipse, Android, ROS etc.
  - Similar to SPLs, but...
    - **Multiple** contributors
    - **Frequent (unsynchronized) evolution** of variable assets

- Variability in Space
  - Configuration
    - **Variant** of system

- Variability in Time
  - Evolution
    - **Version** of system
Running Example: TurtleBot Driver

Cross-Tree Constraints
1. Autonomous → Detection
2. Keyboard ∨ Gamepad → Webservice
Definition 1: Feature Model

A feature model is a 4-tuple \( FM = (\mathcal{F}, \prec, \lambda, \Phi) \) with

1. \( \mathcal{F} \): a finite set of features
2. \( \prec \subseteq \mathcal{F} \times \mathcal{F} \): a decomposition relation on \( \mathcal{F} \) forming a rooted tree of features
3. \( \lambda : \mathcal{P}(\mathcal{F}) \rightarrow \mathbb{N}_0 \times \mathbb{N}_0 \): a function assigning minimum and maximum cardinality to features and feature groups
4. \( \Phi \): a set of propositional formulas over \( \mathcal{F} \) representing cross-tree constraints

Cross-Tree Constraints

1. Autonomous \( \rightarrow \) Detection
2. Keyboard \( \lor \) Gamepad \( \rightarrow \) Webservice
Definition 2 (Feature Model Semantics). A configuration $C \subseteq \mathcal{F}$ of a feature model has to satisfy the following conditions to be valid:

1. Cardinality constraints of features and feature groups are respected by the configuration.
   \[ f \in C \land f \prec F \land \lambda(F) = (k, l) \Rightarrow k \leq |F \cap \mathcal{F}| \leq l \]

2. For each selected feature, the parent feature has to be in the configuration.
   \[ f_1 \in C \land f_2 \in \mathcal{F} \land f_2 \prec f_1 \Rightarrow f_2 \in C \]

3. All propositional formulas of the cross-tree constraints have to be satisfied by the configuration.
   \[ C \models \bigwedge_{\phi_i \in \Phi} \phi_i \]
Problem: No Variability in Time in Feature Models

- Constraints caused by evolution (excerpt)
  - iClebo Kobuki is incompatible with TurtleBot v1.0
  - iClebo Kobuki requires at least TurtleBot v2.0

- Problem
  - Feature model captures only variability in space
  - Only exactly one version of variable asset in feature model (no variability in time)
Requirements on Solution

- **Requirement 1**
  Support for feature versions as units of configuration.

- **Requirement 2**
  Specification of the relation of versions in logical development lines (i.e., predecessor and successor versions including branching).

- **Requirement 3**
  Expression of dependencies on and incompatibilities with version ranges.

- **Requirement 4**
  Support for users in selecting suitable combinations of versions from a selection of features.
Contribution 1

Hyper Feature Models (HFMNs)
Definition 3: Hyper Feature Model (HFM)

1. $\mathcal{F}$: a finite set of features
2. $\mathcal{V}$: a finite set of feature versions
3. $\preceq \subseteq \mathcal{F} \times \mathcal{F}$: a decomposition relation on $\mathcal{F}$ forming a rooted tree of features
4. $\lambda: \mathcal{P}(\mathcal{F}) \rightarrow \mathbb{N}_0 \times \mathbb{N}_0$: a function assigning minimum and maximum cardinality to features and feature groups
Definition 5: HFM Semantics

Variability in Space and Time with Hyper Feature Models

Seidl, Schaefer, Aßmann

22.01.2014

Ultrasound

1.0

0.9

0.8

Infrared

2.2

2.0

1.0

Bump

1.0

Autonomous

2.0

1.0

1.1

Gamepad

2.0

1.0

1.1

Engine

Create 1.2

1.0

1.1

Kobuki 1.0

Movement

1.0

1.1

1.2

2.0

[1..1]

Webservice

1.0

1.1

Detection

1.0

1.1

[1..3]

Keyboard

1.0

Gamepad

1.0

2.0

TurtleBot

1.0

1.1

2.0

2.1

Autonomous

1.0

2.0

1.1

2.2

Infrared

1.0

2.0

2.2

Ultrasound

0.8

0.9

1.0
**Definition 5** (Hyper Feature Model Semantics). A configuration $C \subseteq F \cup \mathcal{V}$ of a Hyper Feature Model has to satisfy the following conditions to be valid:

1. Cardinality constraints of features and feature groups are respected by the configuration.
   
   $f \in C \land f \prec F \land \lambda(F) = (k, l) \Rightarrow k \leq |F \cap F| \leq l$

2. For each selected feature, the parent feature has to be in the configuration.
   
   $f_1 \in C \land C \land f_2 \in F \land f_2 \prec f_1 \Rightarrow f_2 \in C$

3. For each selected version, the containing feature also has to be part of the configuration.

   $\forall v \in C \land C \land \mathcal{V}(f) = \mathcal{V}_f \land v \in \mathcal{V}_f \rightarrow f \in C$

4. For each selected feature, there has to be exactly one version in the configuration.

   $\forall f \in C \land C \land \mathcal{V}(f) = \mathcal{V}_f \rightarrow \exists v \in \mathcal{V}_f : v \in C \land (v_1, v_2 \in C \land v_1 = v_2)$
Contribution 2

Version-Aware Constraint Language
Definition 6 (Version-Aware Constraint Syntax). Let $f \in F$ be a feature, $v_a, v_b \in V$ versions and $op \in \{>, \geq, =, \leq, <\}$: $V \rightarrow P(V)$ a set of operators over versions. The following are basic expressions of the version-aware constraint language:

1. $f$ (feature presence)
2. $f [v_a - v_b]$ (version range restriction)
3. $f [op v_a]$ (relative version restriction)
4. $?f [v_a - v_b]$ (conditional version range restriction)
5. $?f [op v_a]$ (conditional relative version restriction)

Let $\psi_1$ and $\psi_2$ be version-aware constraints. The following are compound expressions of the version-aware constraint language:

6. $\psi_1$ (basic expression)
7. $(\psi_1)$ (nested expression)
8. $\neg \psi_1$ (negation)
9. $\psi_1 \land \psi_2$ (conjunction)
10. $\psi_1 \lor \psi_2$ (disjunction)
11. $\psi_1 \rightarrow \psi_2$ (implication)
(Conditional) Version Range Restrictions

\[ f [v_a - v_b] \]

?\(f [v_a - v_b] \equiv (f \rightarrow f [v_a - v_b])\)

- **Examples**
  - **TurtleBot [1.0 – 1.1] → Engine [≤ Create 1.2]**
  - ?**Movement [1.2 – 2.0]**

- **Semantics**

- **Usage**
  - Closed intervals (fixed even if evolved)
  - E.g., dependencies of Eclipse manifest files
(Conditional) Relative Version Restrictions

\[ f \left[ op \ v_a \right] \]
\[ \exists f \left[ op \ v_a \right] \equiv (f \rightarrow f \left[ op \ v_a \right]) \]

- Operators: \( op = \{>, \geq, =, \leq, <\} \)

- Examples
  - TurtleBot \([1.0 - 1.1] \rightarrow \text{Engine} \[\leq \text{Create 1.2}\]
  - \(?\text{Engine} \[\geq \text{Kobuki 1.0}\]

- Semantics

- Usage
  - No need to know all versions in the interval explicitly
  - Open intervals (extended in case of evolution)
Contribution 3

Automatic Version Selection
Version-Aware Constraints
1. Autonomous $\rightarrow$ Detection
2. Keyboard $\lor$ Gamepad $\rightarrow$ Webservice
3. Infrared $\geq 2.0$ $\lor$ Ultrasound $\rightarrow$ Detection $\geq 1.1$
4. TurtleBot $\geq 2.0$ $\rightarrow$ Engine $\geq$ Kobuki 1.0
5. TurtleBot $[1.0 - 1.1]$ $\rightarrow$ Engine $\leq$ Create 1.2
6. TurtleBot $\geq 2.0$ $\rightarrow$ ?Webservice $\geq 1.1$

Constraint Satisfaction Problem (CSP) Solver

Scoring to find "best" configuration
Scoring to Find "Best" Configuration

<table>
<thead>
<tr>
<th>Requirement 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>i(v) = 1/(1 + 1) = 0.5</td>
</tr>
<tr>
<td>n(v) = 1/(0 + 1) = 1.0</td>
</tr>
<tr>
<td>s(v) = i(v) * n(v) = 0.5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Feature</th>
<th>Movement</th>
<th>Webservice</th>
<th>Detection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Engine</td>
<td>Create 1.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.0</td>
<td>1.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Kobuki 1.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Keyboard</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gamepad</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Autonomous</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bump</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Infrared</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ultrasound</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

importance(v) = 1/(0 + 1) = 1.0
novelty(v) = 1/(1 + 1) = 0.5
score(v) = i(v) * n(v) = 0.5

TurtleBot [≥ 2.0] → Engine [≥ Kobuki 1.0]
Conclusion

- **Problem:** No variability in time in Feature Models

- **Solution**
  - Hyper Feature Models (HFM)s
  - Version-Aware Constraint Language
  - Automatic Version Selection

- Fulfills initially posed requirements
- Evaluated in case study using TurtleBot driver

- Capturing variability in space and time in Hyper Feature Models
Thank you for your attention!

Questions, Comments, Feedback?