Software Product Line Edits

Leonardo Passos
(lpassos@gsd.uwaterloo.ca)

July 2012
Concepts
Feature Model (FM)

- Hierarchical notation for expression variability in the problem space.

- Notations:
  - FODA
  - Cardinality-based
  - Attribute-based

FODA notation will be used throughout this presentation.
### Configuration Knowledge (CK)

<table>
<thead>
<tr>
<th>PL. Exp.</th>
<th>Asset Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>( e_1 )</td>
<td>( n_{11} \ldots n_{1m} )</td>
</tr>
<tr>
<td>( \vdots )</td>
<td>( \vdots )</td>
</tr>
<tr>
<td>( e_k )</td>
<td>( n_{1k} \ldots n_{1j} )</td>
</tr>
</tbody>
</table>

**Definition**

The semantics of a CK \( K \), denoted \( \llbracket K \rrbracket \), is a function that maps product configurations into a set of asset names.

\[
\llbracket K \rrbracket c = \bigcup_{(e, ns) \in K} ns \quad \text{for} \quad \text{eval}(e, c) = T
\]
Asset Mapping (A)

<table>
<thead>
<tr>
<th>Asset Name</th>
<th>Asset</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n_{11}$</td>
<td>class $X$ {...}</td>
</tr>
<tr>
<td>$\cdot$</td>
<td>$\cdot$</td>
</tr>
<tr>
<td>$n_{1j}$</td>
<td>class $Y$ {...}</td>
</tr>
</tbody>
</table>
Definition
The semantics of a feature model $F$, denoted as $\llbracket F \rrbracket$, is the set of all product configurations (set of feature names) of $F$.

Definition
A tuple $(F, A, K)$, where $F$ is a FM, $A$ an asset mapping and $K$ a CK, is a SPL when:

- the expressions in $K$ only refer to features in $F$
- the asset names in $K$ refer only to the domain of $A$
- $\forall c \in \llbracket F \rrbracket. \{A(n) \mid n \in \llbracket K \rrbracket c\}$ is a valid product
Evolutionary Operations & Mechanics
Assumptions

- Given a SPL $(F, A, K)$, the following is assumed:
  - all propositional logic expressions in $K$ are satisfiable
  - there are no dead features
Transformation Scheme Abstraction
Optional Feature to Mandatory Feature (1)
Optional Feature to Mandatory Feature (2)

where

\[
K' = \{(e', ns) \mid (e, ns) \in K \land e' = \sigma(e_{[P/F]}) \land SAT(e')\}
\]

\[
CTC' = \{e' \mid e \in CTC \land e' = \sigma(e_{[P/F]}) \land e' \neq T\}
\]

\(\sigma\) is a simplification function over Boolean formulae
Mandatory Feature to Optional Feature

A
P
K
F
CTC

A
P
K
F
CTC

⤳
Feature Removal (1)

where

$$A' = \{ (n, a_s) | \exists (e, ns) \in K' \land n \in ns \land (n, a_s) \in A \}$$

$$K' = \{ (e', ns) | (e, ns) \in K \land e' = \sigma(e[\bot/F]) \land SAT(e') \}$$

$$CTC' = \{ e' | e \in CTC \land e' = \sigma(e[\bot/F]) \land e' \neq T \}$$
Feature Removal (2)

\[
\begin{align*}
A' &= \{(n, a_s) \mid \exists (e, ns) \in K' \land n \in ns \land (n, a_s) \in A\} \\
K' &= \{(e', ns) \mid (e, ns) \in K \land e' = \sigma(e[\bot/F]) \land SAT(e')\} \\
CTC' &= \{e' \mid e \in CTC \land e' = \sigma(e[\bot/F]) \land e' \neq T\}
\end{align*}
\]
Feature Removal (3)

where (as before)

\[
A' = \{ (n, a_s) \mid \exists (e, ns) \in K' \land n \in ns \land (n, a_s) \in A \} \\
K' = \{ (e', ns) \mid (e, ns) \in K \land e' = \sigma(e_{[\bot/F]}) \land SAT(e') \} \\
CTC' = \{ e' \mid e \in CTC \land e' = \sigma(e_{[\bot/F]}) \land e' \neq T \}
\]
Merge Features (1) I

(merge $F_1$ and $F_2$ and keep $F_1$)

where

$$K' = \{(e', ns) \mid (e, ns) \in K \land e' = \sigma(e[F_1/F_2]) \land SAT(e')\}$$

$$CTC' = \{e' \mid e \in CTC \land e' = \sigma(e[F_1/F_2]) \land e' \neq T\}$$
Merge Features (1) II

- Note that the asset mapping was not changed (unlikely in practice)

- Too fuzzy to be captured and not useful in practice (automatic tool support):

\[
\begin{align*}
\text{CTC}' & = \{ e' \mid e \in \text{CTC} \land e' = \sigma(e_{[F_1/F_2]}) \land e' \neq T \} \\
\text{K}' & = \{ (e', \{ n'_1, \ldots, n'_p \}) \mid (e, \{ n_1, \ldots, n_p \}) \in \text{K} \land \\
& \quad \forall 1 \leq i \leq p. n_i \rightarrow n'_i \land e' = \sigma(e_{[F_1/F_2]}) \land \text{SAT}(e') \} \\
\text{A}' & = \{ (n', \text{as}') \mid \exists (e, \text{ns}) \in \text{K}. n \in \text{ns} \land n \rightarrow n' \land \\
& \quad \text{A}(n) \rightarrow\rightarrow \text{as}' \}
\end{align*}
\]

\(\rightarrow\rightarrow\): renaming, \(\rightarrow\rightarrow\): evolution

- Therefore, A will be taken as unchanged (needs to be updated manually)
Merge Features (2)

(merge $F_1$ and $F_2$ and keep $F_1$)

where (as before)

\[
K' = \{(e', ns) \mid (e, ns) \in K \land e' = \sigma(e_{[F_1/F_2]}) \land SAT(e')\}
\]

\[
CTC' = \{e' \mid e \in CTC \land e' = \sigma(e_{[F_1/F_2]}) \land e' \neq T\}
\]
(merge $F_1$ and $F_2$ and keep $F_1$)

where (as before)

$$K' = \{(e', ns) \mid (e, ns) \in K \land e' = \sigma(e_{[F_1/F_2]}) \land SAT(e')\}$$
$$CTC' = \{e' \mid e \in CTC \land e' = \sigma(e_{[F_1/F_2]}) \land e' \neq T\}$$
Merge Features (4)

(merge $F_1$ and $F_2$ and keep $F_1$)

where (as before)

$$K' = \{(e', ns) \mid (e, ns) \in K \land e' = \sigma(e_{[F_1/F_2]}) \land SAT(e')\}$$

$$CTC' = \{e' \mid e \in CTC \land e' = \sigma(e_{[F_1/F_2]}) \land e' \neq T\}$$
Merge Features (5)

\[(merge \, F_1 \, and \, F_2 \, and \, keep \, F_1)\]

where (as before)

\[
K' = \{(e', ns) \mid (e, ns) \in K \land e' = \sigma(e_{[F_1/F_2]}) \land SAT(e')\}
\]

\[
CTC' = \{e' \mid e \in CTC \land e' = \sigma(e_{[F_1/F_2]}) \land e' \neq T\}
\]
(merge $F_1$ and $F_2$ and keep $F_1$)

where (as before)

$$K' = \{(e', ns) \mid (e, ns) \in K \land e' = \sigma(e_{[F_1/F_2]}) \land SAT(e')\}$$

$$CTC' = \{e' \mid e \in CTC \land e' = \sigma(e_{[F_1/F_2]}) \land e' \neq T\}$$
Research Directions
Proposal

- Study edit patterns in evolving SPLs
- Extract basic transformations that could be composed to form complex transformations
- Present them and their mechanics in an abstract way (as performed here)
- Assess their frequency in different SPLs from different domains
Envisioned Contribution

- Many edit operations are possible! (Which ones are used in practice???)
- Assessment of pragmatics of SPL evolution
- Allows tool implementers to have a base of which operations to support
Questions?