Software Reuse and Reusability based on Requirements: Product Lines, Cases and Feature-Similarity Models

Hermann Kaindl, Mike Mannion

Structure

- Introduction
- Part 1: Motivation for Retrieving Similar Products in Software Product Lines
- Part 2: Feature Model Based Development
- Part 3: Case-Based Reasoning
- Part 4: Similarity Matching in Software Product Line Development
- Summary and Conclusion
Part 1

- Motivation for Retrieving Similar Products in Software Product Lines

What Products to Build?

Mission ➔ Product Portfolio ➔ Roadmap & Priorities ➔ Product Management ➔ Pricing ➔ Product Marketing ➔ Channels ➔ Sales & Marketing

© Hermann Kaindl, Mike Mannion
Matching Supply & Demand (Kano Model)

Differentiators (Delighters)
Qualifiers (Satisfiers)

Don’t Need – Don’t Care (Indifferent)
Don’t Need – Don’t Want (Reverse)

Customer Perspective
Supplier Perspective

Matching Supply & Demand

Problem domain
Solution domain

Don’t Need, Don’t Get
Not Available

Customer Perspective
Supplier Perspective
Matching Supply & Demand - Feature Evolution

Feature Type - Customer Perspective

- Differentiators
- Qualifiers
- Don’t Need

Feature Type - Supplier Perspective

- Common
- Shared/Unique
- Redundant

Other examples in which a feature’s variability changes over time?

Supply Trends – Sales-Driven Product Optimization ("Feature Creep")
Domain Platform Stretching

- Boeing 737 divided into 3 platforms:
  - Initial (100,200)
  - Classic (300,400,500)
  - NextGen (600,700,800,900)

- Boeing 777 also been designed to be stretched

Motivation - Similarity
The Need for Similarity Support

- Compare existing products against partial target specification
- Compare similar features in different products
- Compare customer's perspective against supplier's perspectives
- Create a new product from ideas in other products

What methods do we have?
Part 2

- Feature Model Based Development

Supply Trends - Product Lines
Supply Trends – Product Line Development Framework

Supply Trends - Product Line Engineering

© Hermann Kaindl, Mike Mannion
Free Selection

- Free selection means allowing a single system requirements engineer (user) to browse a product line model and simply copy and paste a single requirement from anywhere in the model to the single system model.
Problems of Free Selection

- Selecting a single requirement is often not sufficient.
- Random choice can mean illegal choice
  - e.g. 2 mutually exclusive requirements
  - e.g. not choosing generic requirement.
- Untenable number of choices.
- BUT engineers like freedom of choice!!!

Constraint-Based Selection

Common Points

Variation Points
List of Alternatives

Variation Points
Mutual Exclusion

Variation Points
Option
Example

Common Points
- R1.1
- R1.2

Mutual Exclusion
- R2
- R2.1
- R2.2

List of Alternatives
- R1
- R2.1.1
- R2.1.2
- R2.1.3
- R2.2.1
- R2.2.2

Option
- R2.1.3.1

Common Points
- R1.1
- R1.2

Mutual Exclusion
- R2
- R2.1
- R2.2

List of Alternatives
- R1
- R2.1.1
- R2.1.2
- R2.1.3
- R2.2.1
- R2.2.2

Option
- R2.1.3.1
Product Line Model using Formal Representations

- For a product line model \( P \) of product line requirements a logical expression can be defined as
  \[
  E(P) = \{ T_1 \land T_2 \land \ldots \land T_n \mid \{ T_i = a_{i1} \; R_{i1} \; a_{i2} \; R_{i2} \; a_{i3} \; R_{i3} \; \ldots \; R_{i(n-1)} \} \}
  \]
  \[
  a_{in}, \; a_{ij} = s(r_{ij})
  \]
  - where \( r_{ij} \) must be a directly reusable requirement or Variation Point;
  - and \( R_{ij} \in \{ R_{\text{common}}, R_{\text{mutex}}, R_{\text{list_alts}}, R_{\text{option}} \} \}

Mobile Phone Example

- **Common:** There shall be an address book facility
  - Add to address book
  - Search address book
  - Delete from address book

- **Option:** Email facility
  - POP
  - IMAP
  - SMTP

- **Mutual Exclusion:** The mobile phone shall have a display
  - Black and White
  - Colour

- **List of Alternatives:** There shall be the facility to make a phone call by:
  - Dialling number on numeric keypad
  - Pressing memory recall button
  - Pressing ringback button

- **List of Alternatives:** Email protocol
Mobile Phone Example

Mutual Exclusion:
The mobile phone shall have a display
Black and White
Colour

List of Alternatives:
There shall be the facility to make a phone call by:
Dialling number on numeric keypad
Pressing memory recall button
Pressing ringback button

Option: Email facility
List of Alternatives: Email protocol
POP IMAP SMTP

Common:
There shall be an address book facility
Add to address book
Search address book
Delete from address book

Black and White
Colour

Mutual Exclusion:
The mobile phone shall have a display
Black and White
Colour

List of Alternatives:
There shall be the facility to make a phone call by:
Dialling number on numeric keypad
Pressing memory recall button
Pressing ringback button

Option: Email facility
List of Alternatives: Email protocol
POP IMAP SMTP

Common:
There shall be an address book facility
Add to address book
Search address book
Delete from address book
Mobile Phone Example

Common:
The mobile phone shall have a display

Black and White

Dialling number on numeric keypad

Colour

Pressing memory recall button

Pressing ringback button

Option: Email facility

List of Alternatives:

Email protocol

POP

IMAP

SMTP

Mutual Exclusion:
The mobile phone shall have a display

List of Alternatives:

There shall be the facility to make a phone call by:

Dialling number on numeric keypad

Pressing memory recall button

Pressing ringback button

Option: Email facility

List of Alternatives:

Email protocol

POP

IMAP

SMTP
Product Line Model using Formal Representations

- For a product line model $P$ of product line requirements a logical expression can be defined as
  
  $$E(P) = \{T_1 \land T_2 \land ... \land T_n \mid \{T_i = a_{i1} \circ r_{i1} a_{i2} \circ r_{i2} a_{i3} \circ r_{i3} ... a_{i(n-1)} \circ r_{i(n-1)}\}$$

- $a_{ij}, r_{ij}$ must be a directly reusable requirement or Variation Point;
- and $r_{ij} \in \{r_{\text{common}}, r_{\text{mutex}}, r_{\text{list_alts}}, r_{\text{option}} \}$

Mobile Phone Example – Valid Selection

Propositional Logic Expression Evaluates to TRUE or FALSE depending on selections made.

New Product $= T_1 \land T_2 \land T_3 \land T_4$

- $((\text{AddressBook} \land (\text{Add} \land \text{Search} \land \text{Delete})) \land \text{Display} \land (\text{B&W} \lor \text{Colour})) \land \text{CallTypes} \land (\text{Keypad} \lor \text{Memory} \lor \text{Ringback} \lor \text{Voice}) \land \text{Email} \leftrightarrow (\text{EmailProtocol} \land (\text{POP} \lor \text{IMAP} \lor \text{SMTP})))$

i.e.

- $((\text{TRUE} \land (\text{TRUE} \land \text{TRUE} \land \text{TRUE})) \land \text{TRUE} \land \text{FALSE} \land \text{FALSE}) \land \text{TRUE} \land (\text{TRUE} \lor \text{FALSE} \lor \text{TRUE} \lor \text{TRUE}) \land \text{TRUE} \land (\text{TRUE} \lor \text{FALSE} \lor \text{TRUE} \lor \text{TRUE})$ which evaluates to TRUE
Mobile Phone Example – Invalid Selection

New Product = \( T_1 \land T_2 \land T_3 \land T_4 \)

\[
\begin{align*}
(\text{AddressBook} \land (\text{Add} \land \text{Search} \land \text{Delete})) & \land (\text{Display} \land (\text{B&W} \lor \text{Colour}))) & \land (\text{CallTypes} \land (\text{Keypad} \lor \text{Memory} \lor \text{Ringback} \lor \text{Voice})), \\
(\text{Email} \leftrightarrow (\text{EmailProtocol} \land (\text{POP} \lor \text{IMAP} \lor \text{SMTP})))
\end{align*}
\]

i.e.

\[
\begin{align*}
(\text{TRUE} \land (\text{TRUE} \land \text{TRUE} \land \text{FALSE})) & \land (\text{TRUE} \land (\text{TRUE} \lor \text{FALSE})), \\
(\text{TRUE} \land (\text{TRUE} \lor \text{FALSE} \lor \text{TRUE} \lor \text{FALSE})) & \land (\text{TRUE} \leftrightarrow (\text{TRUE} \land (\text{TRUE} \lor \text{FALSE} \lor \text{TRUE}))),
\end{align*}
\]

which evaluates to \text{FALSE because } T_1 \text{ is FALSE}

Variation Point-Based Selection

- Use tree structure and Variation Points to direct requirements selection.
- Start at one of the roots.
- Traverse depth first.
- Ask user to make a choice at each Variation Point.
- Common requirements are automatically selected if their parents are already selected or if they are a root node.
Automated Feature Model Analysis

- Does the configured product satisfy the feature model constraints?
- How many product configurations (if any) satisfy these constraints?
- How many products satisfy a given set of features?
- Any anomalies in the feature model e.g. contradictions, redundancy?
- To what degree (expressed as a numeric value) has a feature model variable or common features.

Feature models (regardless of notation) not suited to compare similarity.


Part 3

- Case-Based Reasoning
Case-Based Reasoning


CBR Characteristics

<table>
<thead>
<tr>
<th>Task</th>
<th>Issues</th>
</tr>
</thead>
<tbody>
<tr>
<td>RETAIN</td>
<td>Concrete cases or Generalised cases; Central knowledge units or distributed units; Indexed or flat or hierarchical; General and/or domain specific ontologies; Rich information beyond feature vectors</td>
</tr>
<tr>
<td>RETRIEVE</td>
<td>Use 1 or more similarity metrics e.g. K nearest Neighbour; Guided or not by deep model of general knowledge; Sequentially or in parallel</td>
</tr>
<tr>
<td>REUSE</td>
<td>Facilities to use directly or modified retrieved case</td>
</tr>
<tr>
<td>REVISE</td>
<td>Editing facilities to create a new &quot;solved&quot; case (i) the end user does it OR (ii) automated procedure: risk is that system makes poor judgement, yet it is added to the case-base which becomes progressively degraded.</td>
</tr>
</tbody>
</table>
CBR Example: ReDSseeDS

RE'18 Tutorial Aug. 21, 2018

© Hermann Kaindl, Mike Mannion
RETAIN

Figure 1. Sample scenario written in RSL

Figure 2. Graph for the sample scenario

https://wordnet.princeton.edu/
RETRIEVE

- similarity between words
- similarity between sentences
- Similarity between scenarios (structured graph comparison)
- A single similarity value calculated

Repository of past software cases

Formulate requirements
Transform
Transform
Transform
Query for similar requirements
Import and merge software cases
REUSE & REVISE

- Select one of the better rated software cases.
- Import it to currently developed software case.
- May include design and implementation artefacts, but also requirements and domain descriptions.
- Merge reused case with currently developed one.

<table>
<thead>
<tr>
<th>Task</th>
<th>Issues</th>
</tr>
</thead>
<tbody>
<tr>
<td>RETAIN</td>
<td>Concrete cases or Generalised cases;</td>
</tr>
<tr>
<td></td>
<td>Central knowledge units or distributed units</td>
</tr>
<tr>
<td></td>
<td>Indexed or flat or hierarchical</td>
</tr>
<tr>
<td></td>
<td>General and/or domain specific ontologies</td>
</tr>
<tr>
<td></td>
<td>Rich information beyond feature vectors</td>
</tr>
<tr>
<td>RETRIEVE</td>
<td>Use 1 or more similarity metrics e.g. K nearest Neighbour</td>
</tr>
<tr>
<td></td>
<td>Guided or not by deep model of general knowledge</td>
</tr>
<tr>
<td></td>
<td>Sequentially or in parallel</td>
</tr>
<tr>
<td>REUSE</td>
<td>Facilities to use directly or modified retrieved case</td>
</tr>
<tr>
<td>REVISE</td>
<td>Editing facilities to create a new &quot;solved&quot; case</td>
</tr>
<tr>
<td></td>
<td>(i) the end user does it OR</td>
</tr>
<tr>
<td></td>
<td>(ii) automated procedure: risk is that system makes poor judgement,</td>
</tr>
<tr>
<td></td>
<td>yet it is added to the case-base which becomes progressively degraded.</td>
</tr>
</tbody>
</table>
Part 4

Similarity Matching in Software Product Line Development

<table>
<thead>
<tr>
<th>Feature Modelling</th>
<th>Case-based Reasoning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model Structure Construction</td>
<td>Complex</td>
</tr>
<tr>
<td>Model Content</td>
<td>Detailed, Precise</td>
</tr>
<tr>
<td>Product Derivation</td>
<td>Constrained facilitated automated product derivation</td>
</tr>
<tr>
<td>Costs of making reusable</td>
<td>Substantial</td>
</tr>
<tr>
<td>Benefits for reuse</td>
<td>Facilitates automated product derivation</td>
</tr>
</tbody>
</table>
Feature-Similarity Model


Construction Process

- Construct a feature model.
- Allocate features to products.
- Calculate similarity values between features.
- Calculate similarity values between specific feature combinations in different product specifications.
- Calculate similarity values between entire product specifications.
## Exercise - Similarity in RE

<table>
<thead>
<tr>
<th>Task</th>
<th>Focus</th>
<th>Value of Similarity Matching</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product Line Scoping</td>
<td>Identify features that:</td>
<td>?</td>
</tr>
<tr>
<td></td>
<td>- distinguish the product line</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- important for target market</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Mark product line boundary.</td>
<td></td>
</tr>
<tr>
<td>Domain Engineering</td>
<td>Specify the common and variable domain requirements of agreed product line</td>
<td>?</td>
</tr>
<tr>
<td>Application Engineering</td>
<td>Generate or specify application specific requirements</td>
<td>?</td>
</tr>
</tbody>
</table>
Summary & Conclusion

Motivation for Similarity

- Customers want to personalise
- Suppliers
  - want to be distinctive
  - want to be more efficient
- Match supply and demand
Research Challenges

- How can similarity matching be factored into existing process models for Product Line Scoping, Domain Engineering, and Application Engineering?
- When to compute a similarity between two products: at product definition or on demand
- What are the thresholds for “similar” and for “different”?
- What are the thresholds for analogy anomaly?

Similarity Metrics
- What combinations are worth computing e.g. pearson coefficient, cosine similarity, euclidean distance, k-nearest neighbour algorithm.
- Use caution and prudence - best when used with data from other reference points.
- Be clear on what you are using the metric for, get general agreement in the organization on which metrics to use, and focus on only a few metrics – less is more.

Selected Work of Presenters