Mining Features from the Object-Oriented Source Code of a Collection of Software Product Variants

R. AL-msie’ddeen, A.-D. Seriai, M. Huchard, H. Eyal Salman
LIRMM / CNRS & Montpellier 2 University
Montpellier, France
(Al- msiedee, Abdelhak.Seriai, huchard, eyalsalman)@lirmm.fr

C. Uttado and S. Vauttier
LGI2P / Ecole des Mines d’Al`es
Nîmes, France
(Christelle.Urtado, Sylvain.Vauttier}@mines-ales.fr
Companies usually develop a set of software variants that share some features and differ in other ones to meet specific needs of customers.

To exploit existing software variants to build a software product line (SPL), the first step is to mine the feature model of this SPL. To do so, it is necessary to mine optional and mandatory features.

We propose a new approach to mine features from object-oriented source code based on Formal Concept Analysis and Latent Semantic Indexing. To validate our approach, we applied it on ArgoUML and Mobile Media case studies.
INTRODUCTION: Software Product Variants

Software variants are developed by ad-hoc reuse techniques such as clone-own technique.

Software product variants represent a starting point to build software product line (SPL).

To exploit these software product variants for Systematic Reuse, like Software Product Line development, it is necessary to detect the common and the optional features among these variants.
INTRODUCTION: Feature and Feature Model

- A Feature is a system property relevant to some stakeholder used to capture commonalities or variations among systems in a family [CLEMENTS 01].

- Feature models (FMs) are tree-like graphs of features and relationships among them. FMs in SPLE are used to represent commonality and variability of SPL members at different levels of abstraction [CLEMENTS 01].

![Text Editor variants Feature Model](image)
MOTIVATIONS:

- Manual reverse engineering of the feature model for the existing software variants is time-consuming, error-prone, and requires substantial effort. So we propose a new approach to mine features from a collection of software product variants source code based on Formal Concept Analysis (FCA) and Latent Semantic Indexing (LSI).
FEATURE MINING APPROACH: The Main Hypotheses

- The general objective of our approach is to extract FM which model common and variable features of software product variants.

- We focus our work on the identification of functional features. Functional features express the behavior or the way users may interact with a product.

- We assume that the functional features are implemented at the programming language level. Thus, the elements of the source code reflect these features.
Feature Mining Approach: The Main Hypotheses

- In our work each software product variant \( P_n \) is abstracted as a set of Object-oriented Building Elements (OBEs).

- Object-oriented Building Elements = package, class, attribute, method, local variable, attribute access, method invocation.

- We consider that a feature corresponds to one and only one set of OBEs. So the feature always has the same implementation in all product variants where it is present.

- We also consider that feature implementations may overlap: a given OBE can be shared between several features’ implementations. In our approach, we name such shared OBE as a junction.
FEATURE MINING PROCESS: Optional Feature

- As a feature corresponds to one and only one set of OBEs, then the optional feature is implemented by the same set of Variable OBEs in all software product variants where it is present.

- We define Block Of Variation (BV) as a set of Variable OBEs which are always appear together in some software variants or in a single variant.

- The subsets of Variable OBEs that belong to BV and represent one and only one feature are called Atomic Blocks of Variation (ABV).

- Block of Variation (BV) is composed of a set of Atomic Blocks Of Variation (ABV). To determine its various parts (partitions), we rely on the clustering of the closest Variable OBEs based on the lexical similarity (i.e.; textual similarity).
FEATURE MINING PROCESS: Mandatory Feature

- As a feature corresponds to one and only one set of OBEs, then the mandatory feature is implemented by the same set of Common OBEs in all software product variants.

- We define Common Block (CB) as a set of Common OBEs which are always appear together in all software variants.

- The subsets of Common OBEs that belong to the Common Block (CB) and represent one and only one mandatory feature are called Common Atomic Block (CAB).

- Common block (CB) is composed of a set of Common Atomic Blocks (CAB). To determine its various parts (partitions), we rely on the clustering of the closest Common OBEs based on the lexical similarity.
Mining Feature From a Collection of Software Product Variants

R. Almsie'deen et al.

OBE TO Feature Mapping Model:

1. Software Product Variants (PVs)
   - 1..* Product
   - 1..* OBE

2. Variable OBE (VOBE)
   - 1..* Block of variation (BV)
   - 1..* Common OBE (COBE)
   - 1..* Common Block (CB)
   - 1..* Common Atomic block (CAB)

3. Atomic block of variation (ABV)

Approach Elements

Correspondence:
- Software Product Variants (PVs) correspond to Object-oriented Building Elements (OBEs)
- Variable OBE (VOBE) corresponds to Feature Model (FM) Elements

Object-oriented Building Elements (OBEs):
- Package
  - 1..* Class
  - 1..* Attribute
  - 1..* Method

- Attribute Access
- Method Invocation
- Local Variable

Feature Model (FM) Elements:
- Mandatory Feature
  - 1..* Feature Model
  - 0..* Constraint
  - 1 Root
  - 0..* Cross-tree constraint
  - Group of features Constraint
  - 0..*
MAPPING MODEL: Illustrative Example:

Software_1

Software_2

Software_3

Atomic Block of Variation (AB)

Block of Variation (BV)

Common Atomic Block (CAB)

Common Block (CB)

Legend
OBJECT-ORIENTED SOURCE CODE VARIATION ANALYSIS:

1. **Package Variation**
   - Package Set Variation
   - Package Content Variation

2. **Class Variation**
   - Class Signature Variation
     - Access Level → Public, Private, ...
     - Relationship → Inheritance (Superclass), Interface
   - Class Content Variation
     - Methods Set Variation (Name)
     - Attributes Set Variation (Name)

3. **Attribute Variation** → (Access Level, Data Type)

4. **Method Variation**
   - Signature
     - Access Level
     - Returned Data Type
     - Parameters List order & data type
   - Body
     - Local variables
     - Method Invocation
     - Attribute Access
REDUCE THE SEARCH SPACE:

Initial Search Space

- Common Block (CB)
  - Common OBEs

- Variable OBEs

Set of Variable OBEs

Block of Variation (BV)_1

- Atomic Set of Variable OBEs_1
  - Optional Feature_1

Block of Variation (BV)_n

- Atomic Set of Variable OBEs_n
  - Optional Feature_1

Common Block (CB) for all Product Variants (PVs)

- Atomic Set of Common OBEs_1
  - Mandatory Feature_1

- Atomic Block of Variation (ABV)_1
  - Optional Feature_1

- Atomic Block of Variation (ABV)_n
  - Optional Feature_1

Common Atomic Block (CAB)_1

Common Atomic Block (CAB)_n
THE FEATURE MINING PROCESS:

Implementation Space

Software Product Variants

Feature Space

Features

Mandatory Feature

Optional Feature

THE FEATURE MINING PROCESS:

1. **Static Analysis**

   - **OBEs**
     - **Clustering**
     - **Commonalities and Variabilities Computation**

2. **Lexical Similarity Computation**

   - **FCA**
   - **Common Block**
   - **Common OBEs**

   - **Similarity Matrix**
     - **Lexical Similarity Computation**
     - **LSI**
     - **LSI**

   - **Variable OBEs**
     - **FCA**

   - **Clustering**
     - **Lexical Similarity Computation**
     - **LSI**

   - **Common Block**
     - **FCA**

   - **Atomic Block of Variation**
     - **Clustering**
     - **Similarity Matrix**
     - **Lexical Similarity Computation**

   - **Common Atomic Block**
     - **Clustering**
     - **Similarity Matrix**
     - **Lexical Similarity Computation**

   - **FCA**
TECHNIQUES USED FOR CLASSIFICATION:

- Formal Concept Analysis (FCA): [KUZNETSOV 07] [GANTER 99] [ARÉVALO 07].

- Latent Semantic Indexing (LSI): [MARCUS 03].

- The effectiveness of IR methods is measured by their **RECALL**, **PRECISION** and **F-MEASURE**.

\[
\text{Precision} = \frac{\sum_{i} \text{Correctly retrieved links}}{\sum_{i} \text{Total retrieved links}} \%
\]

\[
\text{Recall} = \frac{\sum_{i} \text{Correctly retrieved links}}{\sum_{i} \text{Total relevant links}} \%
\]

\[
F - \text{Measure} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \%
\]

- **Recall** is the percentage of correctly retrieved links (OBEs) to the total number of relevant links (OBEs).

- **Precision** is the percentage of correctly retrieved links (OBEs) to the total number of retrieved links (OBEs).

- **F-measure** is a balanced measure that takes into account both precision and recall.
IDENTIFYING ATOMIC BLOCKS: Identifying the Common Block and Blocks of Variation

TEXT EDITOR VARIANTS DESCRIBED BY THEIR FEATURES

<table>
<thead>
<tr>
<th>Variant Name</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Editor_1</td>
<td>Core (Open, Close, Print)</td>
</tr>
<tr>
<td>Editor_2</td>
<td>Core, Select_all</td>
</tr>
<tr>
<td>Editor_3</td>
<td>Core, Copy, Paste</td>
</tr>
</tbody>
</table>

The Formal Context for the Text Editor Variants:

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Editor_1</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>Editor_2</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>Editor_3</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>
</tbody>
</table>

Approach Input: Object oriented source code of a set of product variants.
The AOC-poset for the Formal Context of Text Editor Variants.
- Similarity between OBEs in BV or CB is described by *Cosine Similarity Matrix*.

- We consider the most widely used *Threshold* for cosine similarity that equals to 0.70.

- The Cosine similarity matrix of Concept_2.

<table>
<thead>
<tr>
<th></th>
<th>Class Copy/Text Copy</th>
<th>Class Paste/Text Paste</th>
<th>Method Copy/Settings Copy/Text</th>
<th>Method Paste/Setting Paste/Text</th>
<th>Package Copy</th>
<th>Package Paste</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class Copy/Text Copy</td>
<td>1</td>
<td>0.0556</td>
<td>0.9921</td>
<td>0.1715</td>
<td>0.9964</td>
<td>-0.0166</td>
</tr>
<tr>
<td>Class Paste/Text Paste</td>
<td>0.0556</td>
<td>1</td>
<td>0.1802</td>
<td>0.9931</td>
<td>-0.0285</td>
<td>0.9973</td>
</tr>
<tr>
<td>Method Copy/Settings Copy/Text</td>
<td>0.9921</td>
<td>0.1802</td>
<td>1</td>
<td>0.2935</td>
<td>0.9780</td>
<td>0.1086</td>
</tr>
<tr>
<td>Method Paste/Setting Paste/Text</td>
<td>0.1715</td>
<td>0.9931</td>
<td>0.2935</td>
<td>1</td>
<td>0.0880</td>
<td>0.9821</td>
</tr>
<tr>
<td>Package Copy</td>
<td>0.9964</td>
<td>-0.0285</td>
<td>0.9780</td>
<td>0.0880</td>
<td>1</td>
<td>-0.1007</td>
</tr>
<tr>
<td>Package Paste</td>
<td>-0.0166</td>
<td>0.9973</td>
<td>0.1086</td>
<td>0.9821</td>
<td>-0.1007</td>
<td>1</td>
</tr>
</tbody>
</table>
IDENTIFYING ATOMIC BLOCKS: Measuring OBEs’ Similarity Based on LSI ➞ Graph
Formal context of Concept_2.

<table>
<thead>
<tr>
<th></th>
<th>Class CopyText Copy</th>
<th>Class PasteText Paste</th>
<th>Method CopySettings CopyText</th>
<th>Method PasteSetting PasteText</th>
<th>Package Copy</th>
<th>Package Paste</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class CopyText Copy</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Class PasteText Paste</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Method CopySettings CopyText</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Method PasteSetting PasteText</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Package Copy</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Package Paste</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Atomic Blocks (i.e., features) Mined from Concept 2.
IDENTIFYING ATOMIC BLOCKS: Mandatory and Optional features for Editor Variants.

Block of Variations

- Class (SelectAllSettings_Editor.SelectAll)
- Package (Editor.SelectAll)

Editor 2

- Block of Variations 2

- Class (CopySettings_CopyText_Editor.Copy)
- Class (CopyText_Editor.Copy)
- Package (Editor.Paste)
- Class (PasteText_Editor.Paste)
- Method (PasteSettings_PasteText_Editor.Paste)

Editor 3

- Block of Variations 1

- Package (Editor.Copy)
- Method (CopySettings_CopyText_Editor.Copy)
- Class (CopyText_Editor.Copy)
- Package (Editor.Paste)
- Class (PasteText_Editor.Paste)
- Method (PasteSettings_PasteText_Editor.Paste)

Common Block

- Package (Editor)
- Package (Editor.Management)
- Class (Open_Editor.Management)
- Class (Close_Editor.Management)
- Class (Print_Editor.Management)

Common Atomic Block = Mandatory Feature

Atomic Block of Variations = Optional Feature
<table>
<thead>
<tr>
<th>Product #</th>
<th>Mobile Media Product Description</th>
<th>LOC</th>
<th>NOP</th>
<th>NOC</th>
<th>NOOBE</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>Mobile photo core</td>
<td>1,046</td>
<td>6</td>
<td>15</td>
<td>822</td>
</tr>
<tr>
<td>P2</td>
<td>Exception handling enabled</td>
<td>1,159</td>
<td>7</td>
<td>24</td>
<td>925</td>
</tr>
<tr>
<td>P3</td>
<td>Sorting and edit photo label enabled</td>
<td>1,314</td>
<td>7</td>
<td>25</td>
<td>1,040</td>
</tr>
<tr>
<td>P4</td>
<td>Favourites enabled</td>
<td>1,363</td>
<td>7</td>
<td>25</td>
<td>1,066</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Product #</th>
<th>ArgoUML Product Description</th>
<th>LOC</th>
<th>NOP</th>
<th>NOC</th>
<th>NOOBE</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>All Features disabled</td>
<td>82,924</td>
<td>55</td>
<td>1,243</td>
<td>74,444</td>
</tr>
<tr>
<td>P2</td>
<td>All Features enabled</td>
<td>120,348</td>
<td>81</td>
<td>1,666</td>
<td>100,420</td>
</tr>
<tr>
<td>P3</td>
<td>Only Logging disabled</td>
<td>118,189</td>
<td>81</td>
<td>1,666</td>
<td>98,988</td>
</tr>
<tr>
<td>P4</td>
<td>Only Cognitive disabled</td>
<td>104,029</td>
<td>73</td>
<td>1,451</td>
<td>89,273</td>
</tr>
<tr>
<td>P5</td>
<td>Only Sequence diagram disabled</td>
<td>114,969</td>
<td>77</td>
<td>1,608</td>
<td>96,492</td>
</tr>
<tr>
<td>P6</td>
<td>Only Use case diagram disabled</td>
<td>117,636</td>
<td>78</td>
<td>1,625</td>
<td>98,468</td>
</tr>
<tr>
<td>P7</td>
<td>Only Deployment diagram disabled</td>
<td>117,201</td>
<td>79</td>
<td>1,633</td>
<td>98,323</td>
</tr>
<tr>
<td>P8</td>
<td>Only Collaboration diagram disabled</td>
<td>118,769</td>
<td>79</td>
<td>1,647</td>
<td>99,358</td>
</tr>
<tr>
<td>P9</td>
<td>Only State diagram disabled</td>
<td>116,431</td>
<td>81</td>
<td>1,631</td>
<td>97,760</td>
</tr>
<tr>
<td>P10</td>
<td>Only Activity diagram disabled</td>
<td>118,066</td>
<td>79</td>
<td>1,648</td>
<td>98,777</td>
</tr>
</tbody>
</table>
### RESULT: FEATURES MINED FROM MOBILE MEDIA AND ARGOUML SOFTWARES

<table>
<thead>
<tr>
<th>Case Study</th>
<th>Feature</th>
<th>Evaluation Metrics</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Mobile Media Features</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Album Management</td>
<td>x</td>
<td>83%</td>
</tr>
<tr>
<td>Splash Screen</td>
<td>x</td>
<td>62%</td>
</tr>
<tr>
<td>Create Album</td>
<td>x</td>
<td>71%</td>
</tr>
<tr>
<td>Delete Album</td>
<td>x</td>
<td>57%</td>
</tr>
<tr>
<td>Create Photo</td>
<td>x</td>
<td>81%</td>
</tr>
<tr>
<td>Delete Photo</td>
<td>x</td>
<td>58%</td>
</tr>
<tr>
<td>View Photo</td>
<td>x</td>
<td>80%</td>
</tr>
<tr>
<td>Exception handling</td>
<td>x</td>
<td>62%</td>
</tr>
<tr>
<td>Edit Photo Label</td>
<td>x</td>
<td>81%</td>
</tr>
<tr>
<td>Favourites</td>
<td>x</td>
<td>63%</td>
</tr>
<tr>
<td>Sorting</td>
<td>x</td>
<td>78%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ArgoUML Features</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Class Diagram</td>
<td>x</td>
<td>72%</td>
</tr>
<tr>
<td>Diagram</td>
<td>x</td>
<td>56%</td>
</tr>
<tr>
<td>Deployment Diagram</td>
<td>x</td>
<td>72%</td>
</tr>
<tr>
<td>Collaboration Diagram</td>
<td>x</td>
<td>100%</td>
</tr>
<tr>
<td>Use Case Diagram</td>
<td>x</td>
<td>74%</td>
</tr>
<tr>
<td>State Diagram</td>
<td>x</td>
<td>67%</td>
</tr>
<tr>
<td>Sequence Diagram</td>
<td>x</td>
<td>100%</td>
</tr>
<tr>
<td>Activity Diagram</td>
<td>x</td>
<td>64%</td>
</tr>
<tr>
<td>Cognitive Support</td>
<td>x</td>
<td>90%</td>
</tr>
<tr>
<td>Logging</td>
<td>x</td>
<td>78%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Mandatory</th>
<th>Optional</th>
<th>Precision</th>
<th>Recall</th>
<th>F-Measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Album Management</td>
<td>x</td>
<td></td>
<td>83%</td>
<td>62%</td>
<td>70%</td>
</tr>
<tr>
<td>Splash Screen</td>
<td>x</td>
<td></td>
<td>71%</td>
<td>57%</td>
<td>63%</td>
</tr>
<tr>
<td>Create Album</td>
<td>x</td>
<td></td>
<td>81%</td>
<td>58%</td>
<td>67%</td>
</tr>
<tr>
<td>Delete Album</td>
<td>x</td>
<td></td>
<td>80%</td>
<td>62%</td>
<td>69%</td>
</tr>
<tr>
<td>Create Photo</td>
<td>x</td>
<td></td>
<td>81%</td>
<td>52%</td>
<td>63%</td>
</tr>
<tr>
<td>Delete Photo</td>
<td>x</td>
<td></td>
<td>78%</td>
<td>63%</td>
<td>69%</td>
</tr>
<tr>
<td>View Photo</td>
<td>x</td>
<td></td>
<td>87%</td>
<td>68%</td>
<td>76%</td>
</tr>
<tr>
<td>Exception handling</td>
<td>x</td>
<td></td>
<td>100%</td>
<td>70%</td>
<td>82%</td>
</tr>
<tr>
<td>Edit Photo Label</td>
<td>x</td>
<td></td>
<td>100%</td>
<td>77%</td>
<td>87%</td>
</tr>
<tr>
<td>Favourites</td>
<td>x</td>
<td></td>
<td>100%</td>
<td>80%</td>
<td>88%</td>
</tr>
<tr>
<td>Sorting</td>
<td>x</td>
<td></td>
<td>100%</td>
<td>78%</td>
<td>87%</td>
</tr>
</tbody>
</table>
RELATED WORK:

- In our previous work [AL-MSIE’DEEN 13] we presented an approach for feature location from the object-oriented source code of a collection of software product variants using Formal Concept Analysis.

- The approach proposed by Ziadi et al. [ZIADI 12] is the closest one to our work. They identify all common features as a single mandatory feature. Moreover, they do not distinguish between optional features that appear together in a set of variants. Also, their approach doesn’t consider the method body.

- Xue et al. [XUE 12] propose an automatic approach to identify the traceability link between a given collection of features and a given collection of source code variants based on the features descriptions.
CONCLUSION:

✓ We proposed an approach based on FCA and LSI to mine features from the object-oriented source code of software product variants.

✓ We have implemented our approach and evaluated its produced results on two case studies. Results showed that most of the features were identified.

✓ The threat to the validity of our approach is that developers might not use the same vocabularies to name OBEs across software product variants. This means that lexical similarity may be not reliable in all cases to identify common and optional features.
In future work, we plan to combine both textual and semantic similarity measures to be more precise in determining feature implementation.

We also plan to use the mined common and variable features to automate the building of the studied software family’s feature model and all constraints between its features.
REFERENCES:


Thank You For Your Attention