Five-Step Automated Analysis of Feature Models
Student Conference on Software Engineering and Database Systems

Reimar Schröter

July 18, 2012
Agenda

1. Software Product Lines
   Feature Models
   Automated Analysis

2. Problem Statement
   Sate-of-the-art Approach of Analysis Tools
   Error Categories

3. Investigation of Analysis-Tools

4. Five-Step Approach for Automated Analysis
Software Product Lines (SPL)

SPL:
Similar programs on a common code base

Feature Models:
Hierarchical arrangement for commonalities and differences of SPL products

http://www.bmwusa.com
Software Product Lines (SPL)

SPL:
Similar programs on a common code base

Feature Models:
Hierarchical arrangement for commonalities and differences of SPL products

see Apel and Kästner [2009]
Software Product Lines (SPL)

SPL:
Similar programs on a common code base

Feature Models:
Hierarchical arrangement for commonalities and differences of SPL products

see Apel and Kästner [2009]
Feature Models

List Base Debug
  Synchronize Sort
  {oneof
    MergeSort
    QuickSort
  }

Synchronize -> MergeSort

VELVET, see Rosenmüller et al. [2011]

Possible Inconsistencies
  e.g.: non-selectable features
  Reason: Crosstree constraints, e.g.: List -> MergeSort

Real-world feature models are very huge, see Batory et al. [2006] → Automated Analysis
Feature Models

```
concept List {
  mandatory feature Base;
  feature Debug;
  feature Synchronize;
  feature Sort {
    oneof {
      feature MergeSort;
      feature QuickSort;
    }
  }
}
constraint Synchronize \rightarrow MergeSort;
```

VELVET, see Rosenmüller et al. [2011]
Feature Models

Possible Inconsistencies
- e.g.: non-selectable features
- Reason: Cross-tree constraints, e.g.: List -> MergeSort

Real world feature models are very huge, see Batory et al. [2006]

→ Automated Analysis

VELVET, see Rosenmüller et al. [2011]
Automated Analysis, see Benavides et al. [2010]

Void Feature Model: SPL represents no product
Automated Analysis, see Benavides et al. [2010]

Void Feature Model: SPL represents no product

Core Features (CF): feature is present in all products
Automated Analysis, see Benavides et al. [2010]

Void Feature Model: SPL represents no product
Core Features (CF): feature is present in all products
False-Optional Feature: feature is non mandatory, but is CF
Automated Analysis, see Benavides et al. [2010]

Void Feature Model: SPL represents no product
Core Features (CF): feature is present in all products
False-Optional Feature: feature is non mandatory, but is CF
Dead Feature: feature is not presented in any product
Automated Analysis - Two-Step Approach

concept List { // l
  ...
  feature Debug; // d
  feature Sort; // s
  ...
}

Translation in propositional formulas

1 ∧ ...
∧ (d → l) ∧ (s → l)
∧ ...

Execution of Analysis

Results

But what happens, if errors are already in the input feature models?

Core Features?
Automated Analysis - Two-Step Approach

![Diagram of feature analysis process]

concept List{ // l
  ...
  feature Debug; // d
  feature Sort; // s
  ...
}

Translation in propositional formulas

l ∧ ... 
∧ (d → l) ∧ (s → l)
∧ ...

Execution of Analysis

Results

But what happens, if errors are already in the input feature models? ➔ Core Features?
Error Categories

**Grammar Errors**, e.g.: Duplicated group definition

```plaintext
concept List{
    feature Sort{
        oneof{
            someof{
                // ...}
        }
    }
}
```

Satisfiability Errors, e.g.: Void feature model

What happens, if it is not detected before further analyses are executed.
Error Categories

**Grammar Errors**, e.g.: Duplicated group definition
→ Which group is used for analyses?

```plaintext
concept List{
  feature Sort{
    oneof{
      someof{
        // ...}
      }
    }
  }
```

**Syntactical Errors**, e.g.: Duplicated feature definition
→ Which feature Synchronize (optional or mandatory) is used.

```plaintext
mandatory feature Synchronize;
feature Debug;
feature Synchronize;
// ...
```

**Satisfiability Errors**, e.g.: Void feature model
→ What happens, if it is not detected before further analyses are executed.

```plaintext
concept List{
  feature Sort{
    oneof{
      feature MergeSort;
      feature QuickSort;
    }
  }
  constraint Sort. MergeSort;
  constraint Sort. QuickSort;
}
```
Error Categories

**Grammar Errors**, e.g.: Duplicated group definition
→ Which group is used for analyses?

```plaintext
concept List{
  mandatory feature Synchronize;
  feature Debug;
  feature Synchronize;
  //...
}
```

**Syntactical Errors**, e.g.:
Duplicated feature definition

```plaintext
concept List{
  feature Sort{
    oneof{
      somehow{
      // ...,
    }
  }
    constraint Sort . Synchronize;
    constraint Sort . Synchronize;
  }
}
```
Error Categories

**Grammar Errors**, e.g.: Duplicated group definition → Which group is used for analyses?

```
concept List{
  mandatory feature Synchronize;
  feature Debug;
  feature Synchronize;
  // ... 
}
```

**Syntactical Errors**, e.g.: Duplicated feature definition → Which feature Synchronize (optional or mandatory) is used.

```
concept List{
  feature Sort{
    oneof{
      someof{
        // ... 
      }
    }
  }
}
```
Error Categories

**Grammar Errors**, e.g.: Duplicated group definition
→ Which group is used for analyses?

```plaintext
concept List{
  mandatory feature Synchronize;
  feature Debug;
  feature Synchronize;
  //...
}
```

**Syntactical Errors**, e.g.: Duplicated feature definition
→ Which feature Synchronize (optional or mandatory) is used.

```plaintext
concept List{
  feature Sort{
    oneof{
      someof{
        // ...
      }
    }
  }
}
```

**Satisfiability Errors**, e.g.: Void feature model

```plaintext
concept List{
  //...
  feature Sort{
    oneof{
      feature MergeSort;
      feature QuickSort;
    }
  }
  constraint Sort.MergeSort;
  constraint Sort.QuickSort;
}
```
Error Categories

**Grammar Errors**, e.g.: Duplicated group definition
→ Which group is used for analyses?

```concept
List{
    mandatory feature Synchronize;
    feature Debug;
    feature Synchronize;
    // ...}
```

**Syntactical Errors**, e.g.: Duplicated feature definition
→ Which feature Synchronize (optional or mandatory) is used.

```concept
List{
    feature Sort{
        oneof{
            someof{
                // ... }
        }
    }
}
```

**Satisfiability Errors**, e.g.: Void feature model
→ What happens, if it is not detected before further analyses are executed.

```concept
// ...
List{
    feature Sort{
        oneof{
            feature MergeSort;
            feature QuickSort;
        }
    }
    constraint Sort.MergeSort;
    constraint Sort.QuickSort;
}
```
Tool Evaluation

**S.P.L.O.T.**

- Set of tools for editing, configuring, analyzing of feature models

  see Mendonca et al. [2009]

**FeatureIDE**

- Supports creating and editing of feature models

  see Thüm et al. [2012]
Tool Evaluation

**S.P.L.O.T.**

- Set of tools for editing, configuring, analyzing of feature models

  see Mendonca et al. [2009]

**FeatureIDE**

- Supports creating and editing of feature models

  see Thüm et al. [2012]

<table>
<thead>
<tr>
<th>Examples</th>
<th>S.P.L.O.T.</th>
<th>FeatureIDE</th>
</tr>
</thead>
<tbody>
<tr>
<td>duplicated group definition</td>
<td>X -contradictory results</td>
<td>X -contradictory results</td>
</tr>
<tr>
<td>duplicated feature names</td>
<td>X -contradictory results</td>
<td>✓-process canceled</td>
</tr>
<tr>
<td>void feature model</td>
<td>✓</td>
<td>X -indirect marked as void</td>
</tr>
</tbody>
</table>
Tool Evaluation

**S.P.L.O.T.**

- Set of tools for editing, configuring, analyzing of feature models

  see Mendonca et al. [2009]

**FeatureIDE**

- Supports creating and editing of feature models

  see Thüm et al. [2012]

<table>
<thead>
<tr>
<th>Examples</th>
<th>S.P.L.O.T.</th>
<th>FeatureIDE</th>
</tr>
</thead>
<tbody>
<tr>
<td>duplicated group definition</td>
<td>-contradictory results</td>
<td>-contradictory results</td>
</tr>
<tr>
<td>duplicated feature names</td>
<td>-contradictory results</td>
<td>✓ -process canceled</td>
</tr>
<tr>
<td>void feature model</td>
<td>✓</td>
<td>X -indirect marked as void</td>
</tr>
</tbody>
</table>

⇒ Two-step approach is insufficient
Five-Step Approach for Automated Analysis

1. **Model-Parsing**
2. **Syntactical Checks**
3. **Model-Translation**
4. **Satisfiability Check**
5. **Analyses**
Five-Step Approach for Automated Analysis

- Parse textual feature model
  - Detect typos, incorrect keyword usage (grammar errors)
  - e.g.: duplicated group definitions
Five-Step Approach for Automated Analysis

- Parse textual feature model
  - Detect typos, incorrect keyword usage (grammar errors)
  - e.g.: duplicated group definitions
- Because of limited expressiveness of grammars
  - Detect Typos in names, incorrect keyword usage (syntactical + grammar errors)
  - e.g.: duplicated feature definition
Five-Step Approach for Automated Analysis

- Parse textual feature model

  - Detect typos, incorrect keyword usage (grammar errors)
  - e.g.: duplicated group definitions

- Because of limited expressiveness of grammars

  - Detect Typos in names, incorrect keyword usage (syntactical + grammar errors)
  - e.g.: duplicated feature definition
Five-Step Approach for Automated Analysis

- Parse textual feature model
  ➔ Detect typos, incorrect keyword usage (grammar errors)
  ➔ e.g.: duplicated group definitions

- Because of limited expressiveness of grammars
  ➔ Detect Typos in names, incorrect keyword usage (syntactical + grammar errors)
  ➔ e.g.: duplicated feature definition

- Check whether the feature model is a void model?
  ➔ Detect not satisfied pre-condition of further analysis
Five-Step Approach for Automated Analysis

- Parse textual feature model
  - Detect typos, incorrect keyword usage (grammar errors)
  - e.g.: duplicated group definitions
- Because of limited expressiveness of grammars
  - Detect Typos in names, incorrect keyword usage (syntactical + grammar errors)
  - e.g.: duplicated feature definition
- Check whether the feature model is a void model?
- Detect not satisfied pre-condition of further analysis
Summary and Conclusion

- Automated Analysis for Feature Models are very important
  - Existing tool support with two-step approach ➔ insufficient (see S.P.L.O.T., FeatureIDE)
Summary and Conclusion

- Automated Analysis for Feature Models are very important
  - Existing tool support with two-step approach ➔ insufficient (see S.P.L.O.T., FeatureIDE)

Introduction of the four-step approach

- Prevents undesired results of the complete development process
- Increases probability of feature model correctness
Thank you for your attention!
Questions?


