Configuration Knowledge of Software Product Lines: A Comprehensibility Study

Elder Cirilo¹, Ingrid Nunes¹,², Alessandro Garcia¹, Carlos Lucena¹

¹Pontifical Catholic University of Rio de Janeiro (PUC-Rio)
²King’s College London
Configurable Product Lines

- Tailored without programming new completion code

```java
01. class X {
02.     #I FDEF TermY
03.     void methodY() {
04.         ...
05.     }
06.     #ENDIF
07.     #I FDEF TermZ
08.     void methodZ() {
09.         ...
10.     }
11.     #ENDIF
```

Feature List
TermY

Preprocessor
Configuration Knowledge

- Focus on two main issues
  - Mapping features to source code
  - Safe composition
- The abstractions are not clearly identifiable when we observe only the source code
- Mappings from feature to a specific abstraction’s source code can cut across many files / places
- The framework’s programming interface defining the desired dependencies between the source code artifacts are not specified in the source languages
Our Study

- Aims at investigating whether the different techniques influence the correct comprehension of the configuration knowledge of framework-based product lines.

- **Techniques**
  - **Annotation** - Mappings from feature to source code are specified into source code.
  
  - **General-purpose modeling** - Specifies the configuration knowledge in one or more general-purpose models.
  
  - **Domain-specific modeling** - Allows the use of domain-specific abstractions in the configuration knowledge specification.
CIDE – Annotation-based approach
pure::variants – General-purpose Modeling
GenArch+ – Domain-specific Modeling

Spring Domain Knowledge Model

- Spring
  - Bean Weather
  - Injection WeatherUserService
    - Bean WeatherUserService
      - Fragment <constructor-arg ref="Weather ..."/>
  - Injection CityDAO
    - Bean CityDAO
      - Fragment <constructor-arg ref="CityDAO ..."/>
      - Fragment <bean id="WeatherService"/>
- Bean Weather UserService
- Bean CityDAO

Implementation Model

- Implementation
  - Source Container src
    - Component br
    - Component olu
    - Component business
    - Component service
      - Class WeatherService.java
      - Class UserService.java
- Resource Container WebContent
  - Folder WEB-INF
    - Folder lib
      - File applicationContext-userservices.xml
      - Fragment <constructor-arg ref="Weather ..."/>
      - Fragment <constructor-arg ref="CityDAO ..."/>

Configuration Model
Experiment Hypotheses

- **H1**: The correct comprehension of the configuration knowledge depends on the different specification techniques.

- **H2**: The time to correctly comprehend the configuration knowledge depends on the different specification techniques.

- **H3**: The individual differences among the expertise of product line engineers do not impact on the correct comprehension of the configuration knowledge.
### Selected Product Lines

- Three independents software product lines implemented on the basis of industry-strength frameworks

- **Characteristic**

<table>
<thead>
<tr>
<th>Domain</th>
<th>e-Shop</th>
<th>OLIS</th>
<th>Buyer</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Features</strong></td>
<td>21</td>
<td>18</td>
<td>12</td>
</tr>
<tr>
<td><strong>Granularity</strong></td>
<td>Coarse and Fine-grained</td>
<td>Coarse-grained</td>
<td>Fine-grained</td>
</tr>
<tr>
<td><strong>Frameworks</strong></td>
<td>Spring-DM, Spring-MVC, iBatis</td>
<td>Struts, Spring, Hibernate</td>
<td>Jadex</td>
</tr>
</tbody>
</table>
Experimental Design

- Our study involved six post-graduate answering three questionnaires, one for each product line following the **Latin Square Design**

  Which abstraction(s)/code asset(s) is(are) related to the feature X?

  How many abstraction(s)/code asset(s) is(are) mapped to the feature Y?

<table>
<thead>
<tr>
<th>Participants</th>
<th>E-Shop</th>
<th>OLIS</th>
<th>Buyer</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1 and P4</td>
<td>G+</td>
<td>PV</td>
<td>C</td>
</tr>
<tr>
<td>P2 and P5</td>
<td>C</td>
<td>G+</td>
<td>PV</td>
</tr>
<tr>
<td>P3 and P6</td>
<td>PV</td>
<td>C</td>
<td>G+</td>
</tr>
</tbody>
</table>
Results

- **Correct Answers and Time Analysis**
  - The correspondence between participant’s number of correct answers and tools/product lines
  - The influence of each approach in the time that each participant spend answering the questionnaire.

- **Expertise Analysis**
  - The influence of participant’s expertise in the number of correct answers
Product Lines vs. Correct Answers

**Buyer** - highest number of correct answers
Lowest number of feature and no diversity of frameworks
**Product Lines vs. Correct Answers**

**OLIS** - intermediate number of correct answers
Well modularized features
**Product Lines vs. Correct Answers**

**E-Shop** - lowest number of correct answers
Features no-well modularized
Techniques vs. Correct Answers

CIDE - lowest number of correct answers in the E-Shop product line
Techniques vs. Correct Answers

**pure::variants** – better number of hits for the E-Shop product line than CIDE
Techniques vs. Correct Answers

**CIDE** – better number of hits for the OLIS product line than pure::variants
Techniques vs. Correct Answers

![Bar Chart]

- Techniques: G+, C, PV, G+
- Correct Answers: G+, C, PV, G+

Legend:
- G+: G+ symbol
- C: C symbol
- PV: PV symbol
- G+: G+ symbol

Data: 567892 ; ; => ? @
### Techniques vs. Time

- Time demanded to answer the questionnaire.

<table>
<thead>
<tr>
<th></th>
<th>G+</th>
<th>PV</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>e-Shop</td>
<td>1:35:47</td>
<td>1:43:29</td>
<td>1:33:45</td>
</tr>
<tr>
<td>OLIS</td>
<td>1:27:51</td>
<td>1:45:42</td>
<td>1:31:09</td>
</tr>
<tr>
<td>Buyer</td>
<td>0:43:05</td>
<td>1:17:42</td>
<td>1:14:42</td>
</tr>
</tbody>
</table>

- Average time to correct answer a question.

<table>
<thead>
<tr>
<th></th>
<th>G+</th>
<th>PV</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0:02:57</td>
<td>0:04:39</td>
<td>0:03:10</td>
</tr>
</tbody>
</table>
## Participant’s Expertise

<table>
<thead>
<tr>
<th></th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P4</th>
<th>P5</th>
<th>P6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spring</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>Struts</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Spring MVC</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>Hibernate</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>iBatis</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>Spring-DM</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Jadex</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td><strong>e-Shop</strong></td>
<td>1.5</td>
<td>1.25</td>
<td>1</td>
<td>2.25</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td><strong>OLIS</strong></td>
<td>3</td>
<td>3</td>
<td>2.75</td>
<td>3.5</td>
<td>1.75</td>
<td>3.25</td>
</tr>
<tr>
<td><strong>Buyer</strong></td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Expertise Results
Expertise Analysis

### Graph Description

- **Legend**:
  - □ 34567/"
  - □ 89:5"
  - □ ;<>+""$
  - ◇ @AB71*C

- **Categories**:
  - G+ (Expertise Category)
  - C (Control Category)
  - PV (Performance Variable)

- **Sections**:
  1. " + - / *01$"
  2. " + - / *01%"
  3. " + - / *01"
  4. " + - / *01#"
  5. " + - / *012"

---

*Note: The diagram visualizes the distribution of expertise (G+) and control (C) across different performance variables (PV) in various categories.*
• There is no relevant difference between the techniques when we **consider the absolute number of correct answers**. However, the time spent for answering them is significantly different;

\[
\begin{array}{c}
H1 \times \\
H2 \checkmark
\end{array}
\]

• The use of domain-specific abstractions in the configuration knowledge specification, in fact, seems to facilitate the understanding of **coarse-grained variability**;

\[
\begin{array}{c}
H1 \checkmark \\
H2 \checkmark
\end{array}
\]
Conclusion

- Because *general-purpose modeling is abstract and hides many relevant details*, it imposes certain restrictions to product line engineers to quickly localize and comprehend the configuration knowledge;

\[ \text{H1 } \checkmark \quad \text{H2 } \checkmark \]

- The *correct comprehension is not associated with individual expertise* about the frameworks that are part of the target product line.

\[ \text{H3 } \checkmark \]
Configuration Knowledge of Software Product Lines: A Comprehensibility Study

Questions?

Elder Cirilo$^1$, Ingrid Nunes$^{1,2}$, Alessandro Garcia$^1$, Carlos Lucena$^1$

$^1$Pontifical Catholic University of Rio de Janeiro (PUC-Rio)
$^2$King’s College London