Feature Interaction Detection in Building Control Systems by Means of a Formal Product Model

Andreas Metzger, Christian Webel
Department of Computer Science
University of Kaiserslautern, Germany

Contents

Motivation

The Product Model Approach

Feature Interaction Detection

Conclusion
Motivation

Complexity of Building Control Systems

Desired Control

Temperature

Ventilation

Occupancy

Humidity

Air Quality

Sound

Light

Coupling of Effects

→ Integrated Building Control System
Motivation

Complexity of Building Control Systems

Problems

• Extension
  → Introduction of Undesirable Relationships

• Reuse
  → Elimination of Required Interrelations

→ Solution: Automatic Detection of Interactions

— The Product Model Approach —
The Product Model Approach

Terms

Product Model
Meta-Model of Development Artefacts and Relations

Formal Product Model
Concise Semantics of Entities
Access to Entities ("Repository")

“Tool Chain”

Artefact Types and Relations

Andreas Metzger – Feature Interaction Workshop – Ottawa, Canada – June, 2003
— Feature Interaction Detection —

Feature Interaction Detection

Scheme
Feature Interaction Detection

Detection at Requirements Level: Concept

"provide required illumination if room is occupied"  
"use daylight to save energy"  
"avoid glare"  
"provide required temperature"

Detection at Requirements Level: Realization

"provide required illumination if room is occupied"  
"use daylight to save energy"  
"avoid glare"  
"provide required temperature"
Feature Interaction Detection
Detection at Requirements Level: Realization

Feature Interaction Detection
Detection at Requirements Level: Realization
Feature Interaction Detection

Detection at Requirements Level: Realization

Feature Interaction Detection

Detection at Object Structure Level: Concept
Feature Interaction Detection

Detection at Object Structure Level: Concept

Signals/Attributes → Coupling of Strategies → Coupling of Tasks

Andreas Metzger – Feature Interaction Workshop – Ottawa, Canada – June, 2003
Feature Interaction Detection

Detection at Strategy Level: Concept

Signals/Attributes → Coupling of Strategies → Coupling of Tasks
Feature Interaction Detection

Detection at Environment Level: Concept

\( \{N1, N2, N3, N4\} @ Env \)

Feature Interaction Detection

Detection at Environment Level: Realization

Building Simulator
Feature Interaction Detection

Detection at Environment Level: Realization

\{N1, N2, N3, N4\} @ \{Tc\}

Feature Interaction Detection

Detection at Environment Level: Realization

\{N1, N2, N3, N4\} @ \{T3, T9\}
--- Conclusion ---

**Conclusion**

**General Applicability**

<table>
<thead>
<tr>
<th>Level of Information</th>
<th>Domain Dependency of Approach</th>
<th>Method Dependency of Approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirements</td>
<td>none</td>
<td>weak (traceability relation)</td>
</tr>
<tr>
<td>Object Structure</td>
<td>strong (strict aggregation, static structure)</td>
<td>weak (aggregation rel., traceability rel.)</td>
</tr>
<tr>
<td>Strategies</td>
<td>weak (reactive systems: signals)</td>
<td>strong (traceability relation)</td>
</tr>
<tr>
<td>Environment</td>
<td>strong (reactive system)</td>
<td>strong (environment simulator)</td>
</tr>
</tbody>
</table>

**Efficiency**

Feature Interaction Detection Tool: ~1900 Lines of Code (Java)
**Conclusion**

- Efficient Approach for FI Detection in Building Control Systems
  - Systematic Mapping of Concepts to Code
  - Abstraction from Development Documents (Product Model)

- Application During System Development
  - Detection of Interactions after Extension/Reuse Activity
  - Metric (“Complexity of System”) → Quality Control

**Perspectives**

- Extension of Application Domain
  - Reactive Systems with Static Structure; e.g., Automotive Control

- Refinement of Detection Concepts
  - Static Analysis of Behavior (Refinement of Product Model)
Results

Case Studies

<table>
<thead>
<tr>
<th>Complexity</th>
<th>Floor32</th>
<th>Floor32X</th>
<th>△</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirements (Needs, Tasks)</td>
<td>285</td>
<td>316</td>
<td>31</td>
</tr>
<tr>
<td>(52, 233)</td>
<td>(64, 252)</td>
<td>(12, 19)</td>
<td></td>
</tr>
<tr>
<td>Control Object Types</td>
<td>37</td>
<td>40</td>
<td>3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Feature Interactions @ Points of Interaction</th>
<th>Floor32</th>
<th>Floor32X</th>
<th>△</th>
</tr>
</thead>
<tbody>
<tr>
<td>at Object Structure Level</td>
<td>32 @ 47</td>
<td>38 @ 53</td>
<td>6 @ 6</td>
</tr>
<tr>
<td>at Environment Level</td>
<td>38 @ 63</td>
<td>44 @ 69</td>
<td>6 @ 6</td>
</tr>
<tr>
<td>△</td>
<td>6 @ 6</td>
<td>6 @ 16</td>
<td></td>
</tr>
</tbody>
</table>

Feature Interaction Detection

Detection at Requirements Level: Implementation

class Main {
    detectInteraction() {
        forall(need ∈ Needs) {
            need.mark(need);
        }
        forall(task ∈ Tasks) {
            if(|task.needSet| > 1 AND |task.realizedRequirement| > 1)
                /* True Point of Interaction identified! */
        }
    }
}

Andreas Metzger – Feature Interaction Workshop – Ottawa, Canada – June, 2003
Feature Interaction Detection

Detection at Requirements Level: Implementation

```java
public class Main {
    void detectInteraction() {
        Iterator allNeeds = /* ... */;
        Need need;
        while(allNeeds.hasNext()) {
            need = (Need)allNeeds.next();
            need.mark(need);
        }
        Iterator allTasks = /* ... */;
        Task task;
        while(allTasks.hasNext()) {
            task = (Task)allTasks.next();
            if(task.getNeedSet().size() > 1 &&
                task.getRealizedRequirement().size() > 1) {
                /* True Point of Interaction identified! */
            }
        }
    }
}
```

The Product Model Approach

Extended Artefact Types and Relations (Part 1)