Towards Context-Aware Feature Modelling using Ontologies

Dennis Wagelaar
System and Software Engineering Lab

Vrije Universiteit Brussel
Contents

➔ Feature Modelling (State-of-the-Art)
➔ A Context Ontology
➔ A Feature Ontology
➔ Expressing External Feature Interactions
Feature Modelling: State-of-the-Art

- Based on FODA
- Numerous extensions to increase expressiveness
  - Multiple feature instances
  - Cardinality constraints
  - Constraint programming/XPath expressions
Feature Modelling: Fragility caused by external factors

- External factor (or context) may be cause for feature interaction
- Change in external factor may require change in feature model
Feature Modelling: Fragility example

Software X

Voice output

Audio cue

Text-to-speech

Audio mixer

Audio hardware

Depends on

Depends on

Depends on

Depends on
Feature Modelling: Proposal

→ Separate external factors
  – Explicitly model external factors in separate context models
  – Explicitly model external factor dependencies in the main feature model
  – Based on ontologies/description logics
A Context Ontology: Introduction

→ Ontologies can serve as a common vocabulary for a domain
  
  – A context ontology can be used as a basis for expressions about context
  
  – Extensions and instantiations can be built on top of the common ontology
A Context Ontology:
Platform-related Concepts

- Modality
  - Rendering Engine
  - Operating System
  - Virtual Machine
  - Middleware
  - Library
- Resource
  - Power Resource
  - Memory Resource
  - CPU Resource
  - Storage Resource
  - Network Resource
  - Network Resource

- Support Modality
- Requires Platform
- Provides Software
- Provides Hardware

Platform
Software
Hardware

A Context Ontology: Java Extension (1)
A Context Ontology: Java Extension (2)
A Feature Ontology: Introduction

Expresses feature modelling domain as an ontology

- Supports modelling of feature dependencies
- Supports definition of valid and invalid configurations
- Supports multiple classification of features
  - (Comparable to Batory’s feature dimensions/algebra)
A Feature Ontology: Concepts

\[ \text{Feature} \]

\[ \text{Configuration} \quad \text{isa} \quad \text{BaseConcept} \]

\[ \text{ValidConfiguration} \quad \text{isa} \quad \text{InvalidConfiguration} \]

\[ \exists \text{transitivelyImpliesFeature} \quad \text{InvalidConfiguration} \]

\[ \text{SoftwareFeature} \quad \equiv \text{context:Software} \]

\[ \text{HardwareFeature} \quad \equiv \text{context:Hardware} \]

\[ \text{ImpliesInvalidConfiguration} \]

\[ \text{impliesFeature}^* \subseteq \text{transitivelyImpliesFeature}^* \]
A Feature Ontology: Instance

```
Network
  └── UserInterface
    └── InstantMessagingClient
        └── io
            └── impliesFeature

feature:Feature
  isa
  └── feature:BaseConcept

  Network
    isa
    └── feature:BaseConcept

  UserInterface
    isa
    └── feature:BaseConcept

  InstantMessagingClient
    isa
    └── feature:BaseConcept

Jabber
  isa
  └── Network

SMS
  isa
  └── Network

SwingGUI
  isa
  └── UserInterface

AWTGUI
  isa
  └── UserInterface

LCDUIGUI
  isa
  └── UserInterface
```
A Feature Ontology: Configuration

- `feature:ValidConfiguration`
- `feature:InvalidConfiguration`

**RequiredFeatures**

- \( \exists \text{transitivelyImpliesFeature} \) InstantMessagingClient
- \( \exists \text{transitivelyImpliesFeature} \) Network
- \( \exists \text{transitivelyImpliesFeature} \) UserInterface

**BothLCDUIandAWT**

- \( \exists \text{transitivelyImpliesFeature} \) LCDUIGUI
- \( \exists \text{transitivelyImpliesFeature} \) AWTGUI

**defaultConfiguration**

- `io = inferred`
- `impliesFeature` instantMessagingClient
- `impliesFeature` jabber
- `impliesFeature` awtGUI
A Feature Ontology: Generative Features Instance
A Feature Ontology: Alternative Classification

```
feature:Feature
  isa
  - CLDCJavaUtilFeature
    isa
    - JavaDataTypes
      io
      javaDataTypes
    - JavaAccessors
      io
      javaAccessors
    - JavaAssocAtt
      io
      javaAssocAtt

impliesFeature

Java2UtilFeature
  isa
  - Java2DataTypes
    io
    java2DataTypes
  - Java2Accessors
    io
    java2Accessors
  - Java2AssocAtt
    io
    java2AssocAtt

impliesFeature
```
External Feature Interactions: Introduction

→ Ontologies can refer to other ontologies
  - A feature can have an *impliesFeature* relationship with a feature in another model
  - Conditions for ValidConfigurations and InvalidConfigurations can be defined in an external model
External Feature Interactions: Example

```
feature:Feature
  isa
  UserInterface
    isa
    SwingGUI
      io
      swingGUI
    AWTGUI
      io
      awtGUI
    LCDUIGUI
      io
      lcdnGUI

java:LCDUI
  isa
  java:AWTEvent
  isa
  java:Swing

java:CLDC
  io
  wtk2.2

java:JDK
  io
  jdk1.5

impliesFeature
```
External Feature Interactions: Configuration

feature:InvalidConfiguration

isa

JDKandCLDC

∃ transitivelyImpliesFeature java:CLDC
∃ transitivelyImpliesFeature java:JDK
Conclusions

➔ Feature model separated from context model
  – Change in external feature interactions does not cause change in main feature model

➔ Ontologies allow for multiple classifications of feature hierarchies
  – Similar to Batory’s work on feature dimensions/algebra
Open issues

- Current feature ontology models feature dependency with instances
  - Features must point directly to context instances, e.g. “jdk1.5” and “wtk2.2”, instead of classes

- Feature ontology depends on context ontol.
  - Given context ontology may not be general enough

- Complete mapping from feature models to ontologies
Questions?

Dennis Wagelaar <dennis.wagelaar@vub.ac.be>
SSEL Website: http://ssel.vub.ac.be/