Verify Feature Models using Protégé-OWL

Hai Wang
The University of Manchester, UK
hwang@cs.man.ac.uk

Yuan Fang Li
National University of Singapore
liyf@comp.nus.edu.sg

Jing Sun
The University of Auckland, New Zealand
j.sun@cs.auckland.ac.nz

Hongyu Zhang
RMIT University, Australia
hongyu@cs.rmit.edu.au

ABSTRACT

Feature models are widely used in domain engineering to capture common and variant features among systems in a particular domain. However, the lack of a widely-adopted means of precisely representing and formally verifying feature models has hindered the development of this area. This paper presents an approach to modeling and verifying feature diagrams using Semantic Web ontologies.

Categories and Subject Descriptors

D.2.13 [**Software Engineering**]: Reusable Software—*Domain engineering*; I.2.4 [**Artificial Intelligence**]: Knowledge Representation Formalisms and Methods—*Representation languages*

General Terms

Languages, Verification

Keywords

Semantic Web, OWL, Ontologies, Feature Modeling

1. INTRODUCTION

Domain engineering, which forms a basis for software product line practices, is a software reuse approach that focuses on a particular application domain. Feature modeling, as "the greatest contribution of domain engineering to software engineering" [1], plays an important role in domain engineering. Quite a number of feature-based reuse approaches have been proposed. However, there is lack of methods and tools that can support automated analysis over a feature model. Such methods and tools should help us check the correctness of a particular feature configuration based on the constraints specified in the feature model.

Semantic Web has emerged as the next generation of the Web since the past few years. We can see that there is a strong similarity between Semantic Web ontology analysis and feature modeling - both of which represent concepts in a particular domain and define how various properties relate them. Hence, we believe that Semantic Web can play important roles in domain engineering.

In this paper, we explore the synergy of domain engineering and Semantic Web. We propose using Semantic Web language and tools to verify feature models in a domain engineering context.

2. OVERVIEW

2.1 Feature modeling

A feature a distinguishable characteristic of a concept that is relevant to some stakeholders [2]. Conceptual relationships among features can be expressed by a feature model as proposed by Kang et al. [4]. A feature model consists of a feature diagram and other associated information (such as rationale, constraints and dependency rules). A feature diagram provides a graphical tree-like notation that shows the hierarchical organization of features. The root of the tree represents a concept node. All other nodes represent different types of features. In a feature model, there are four commonly found feature types – 'mandatory', 'optional', 'alternative and 'or'. A domain can be modeled as a concept.

Feature diagrams sometimes cannot capture all the constraints among the features. We have identified two additional relations among features: 'requires', which means that the presence of some features in a configuration requires the presence of some other features; and 'excludes', which means that the presence of some feature excludes the presence of some other features.

3. FEATURE MODELING USING OWL

In this Section, we describe how to model various feature relations using OWL language constructs. Our presentation of the OWL encoding will be divided into two parts – feature type modeling and feature configuration modeling. The feature modeling in OWL are given in a syntax similar to the "DL syntax" given in [3].

3.1 Conceptual Modeling

Before we model the different feature relations in a feature diagram, we need to build the OWL ontology for the various nodes and edges in the diagram. It was constructed as follows. Each node (concept or feature) in the feature diagram is modeled as an OWL class. And for each of the nodes, we create a `Rule` class. This `Rule` class has two kinds of conditions: **Firstly**, a necessary and sufficient (NS, `EquivalentClass`) condition, using an existential restriction to bind the `Rule` node to the corresponding feature node in the diagram; and **Secondly**, a number of (possibly 0) necessary (N, `subClassOf`) constraints, serving two purposes – to specify how each of its child features are related to this node, capturing the various relations between features and to specify how this feature node is constrained by other features. Lastly, the root concept and features in a feature diagram are inter-related by various feature relations, represented by different edge types in the diagram. In our OWL model, for each of these edges, we create an object-property. We assert that the `range` of the property is the respective feature class.

For a parent feature G and its child features $F_1, ..., F_n$, the initial modeling above produces the following ontology. Note that the symbol ran denotes the range of a property.

$$
\begin{array}{ll}
G \sqsubseteq Thing & hasG \sqsubseteq ObjectProperty \\
GRule \sqsubseteq Thing & \text{ran}\ hasG = G \\
 & GRule \equiv hasG\ \ G \\
F_1 \sqsubseteq Thing & \\
F_1 Rule \sqsubseteq Thing & hasF_1 \sqsubseteq ObjectProperty \\
\cdots & \text{ran}\ hasF_1 = F_1 \\
 & F_1 Rule \equiv hasF_1\ \ F_1 \\
G \neq F_i,\ \text{for } 1 \leq i \leq n & \cdots \\
F_i \neq F_j,\ \text{for } 1 \leq i, j \leq n \wedge i \neq j &
\end{array}
$$

Now we are ready to model the feature relations using the ontology. We use the 'Mandatory' feature type as the example in this paper.

A *mandatory* feature is included if its parent feature is included For each of the *mandatory* features $F_1, ..., F_n$ of a parent feature G, we use one N constraints in $GRule$ to model it. It is a `someValuesFrom` restriction on $hasF_i$, stating that each instance of the rule class must have some instance of F_i class for $hasF_i$. The following ontology fragment shows the modeling of mandatory feature set and parent feature G.

$$GRule \sqsubseteq hasF_1\ \ F_1 \quad \cdots \quad GRule \sqsubseteq hasF_n\ \ F_n$$

Other feature types can be modeled in a similar way.

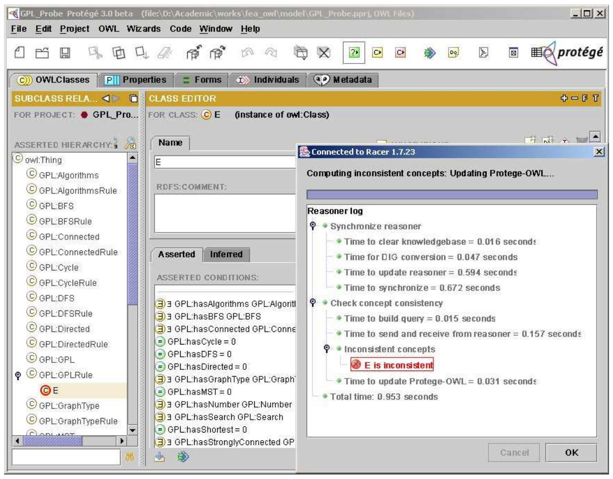
3.2 Verifying Feature Configuration in OWL

A feature configuration is a set of features that an instance of a concept may hold. We model the concept node in the configuration as a subclass of the rule class for the root in a feature diagram and use an existential restriction for each feature included in the configuration. For each feature present in a feature diagram but not in its configuration, we use a "cardinality = 0" restriction to prevent the reasoning engine from inferring the existence of this feature in the configuration. This is necessary because of the Open World Assumption adopted by OWL. We make the concept class to be equivalent (NS condition) to the conjunction of the above constraints.

For a concept instance C derived from a feature diagram with root concept G and a set of features $F1, ..., F_n$, if assuming that $F_1, ..., F_i$ appear in the configuration of C and $F_{i+1}, ..., F_n$ do not, a feature configuration can be modeled as follows.

$$
\begin{aligned}
&C \sqsubseteq GRule \\
&C \equiv \bigsqcap\ (\exists\, hasF_j\ \ F_j,\ for\ 1 \leq j \leq i) \sqcap \\
&\qquad \bigsqcap\ (hasF_k\ = 0,\ for\ i < k \leq n)
\end{aligned}
$$

The feature configuration is constructed as a separate ontology and the reasoning engine is invoked to check its consistency. The configuration is valid if the ontology is checked to be consistent with respect to the feature diagram ontology. Fig. 1 shows that we use Protégé-OWL and RACER to successfully detect the inconsistence in a 'Graph Product Line' (GPL) feature configuration.


GPL_Probe Protégé 3.0 beta (file:\D:\Academic\works\fea_owl\model\GPL_Probe.pprj, OWL Files)
File Edit Project OWL Wizards Code Window Help
protégé
OWLClasses Properties Forms Individuals Metadata
SUBCLASS RELA...
CLASS EDITOR
FOR PROJECT: GPL_Pro...
FOR CLASS: E (instance of owl:Class)
ASSERTED HIERARCHY:
Name
RDFS:COMMENT:
Asserted Inferred
ASSERTED CONDITIONS:
Connected to Racer 1.7.23
Cancel
OK

```
owl:Thing
  GPL:Algorithms
  GPL:AlgorithmsRule
  GPL:BFS
  GPL:BFSRule
  GPL:Connected
  GPL:ConnectedRule
  GPL:Cycle
  GPL:CycleRule
  GPL:DFS
  GPL:DFSRule
  GPL:Directed
  GPL:DirectedRule
  GPL:GPL
  GPL:GPLRule
    E
  GPL:GraphType
  GPL:GraphTypeRule
```

```
E
∃ GPL:hasAlgorithms GPL:Algoritl
∃ GPL:hasBFS GPL:BFS
∃ GPL:hasConnected GPL:Conne
GPL:hasCycle = 0
GPL:hasDFS = 0
GPL:hasDirected = 0
∃ GPL:hasGraphType GPL:Graph
GPL:hasMST = 0
∃ GPL:hasNumber GPL:Number
∃ GPL:hasSearch GPL:Search
GPL:hasShortest = 0
∃ GPL:hasStronglyConnected GP
```

```
Computing inconsistent concepts: Updating Protege-OWL...

Reasoner log
  Synchronize reasoner
    Time to clear knowledgebase = 0.016 seconds
    Time for DIG conversion = 0.047 seconds
    Time to update reasoner = 0.594 seconds
    Time to synchronize = 0.672 seconds
  Check concept consistency
    Time to build query = 0.015 seconds
    Time to send and receive from reasoner = 0.157 seconds
    Inconsistent concepts
      E is inconsistent
    Time to update Protege-OWL = 0.031 seconds
  Total time: 0.953 seconds
```

Figure 1: RACER detects an inconsistency.

4. CONCLUSION

In this paper, we propose a Semantic Web approach for feature modeling and verification. Feature model and configuration verification is an important task in domain engineering. With the growth of the number of features in a feature model, manual checking of validity is very laborious and error-prone. As OWL has a formal and rigorous semantical basis and the decidability of OWL DL, fully automated analysis is achievable. Also as OWL DL reasoning engines are designed to handle large-scale knowledge bases, efficient and effective analysis of large feature models are possible.

5. REFERENCES

[1] K. Czarnecki and U. Eisenecker. *Generative Programming: Methods, Tools, and Applications*. Addison-Wesley, MA., 2000.

[2] M. Simos et al. Software technology for adaptable reliable system (STARS) organization domain modeling (ODM) guidebook version 2.0. Technical Report STARS-VC-A025/001/00, Lockheed Martin Tactical Defense Systems, Manassas, VA, 1996.

[3] I. Horrocks, P. F. Patel-Schneider, and F. van Harmelen. From $\mathcal{SHIQ}$ and RDF to OWL: The making of a web ontology language. *J. of Web Semantics*, 1(1):7–26, 2003.

[4] K. C. Kang, S. Cohen, J. Hess, W. Nowak, and S. Peterson. Feature-oriented domain analysis (FODA) feasibility study. Technical Report CMU/SEI-90TR-21, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, November 1990.