Exploring the Synergies Between Feature Models and Ontologies

Martin Fagereng Johansen[1,2], Franck Fleurey[1]
[1]*SINTEF ICT, Pb. 124 Blindern, 0314 Oslo, Norway,*
{MartinFagereng.Johansen, franck.fleurey}@sintef.no
[2]*Institute for Informatics, University of Oslo,*
Pb. 1080 Blindern, 0316 Oslo, Norway

Mathieu Acher, Philippe Collet, Philippe Lahire
University of Nice Sophia Antipolis,
I3S Laboratory (CNRS UMR 6070),
06903 Sophia Antipolis Cedex, France
{acher,collet,lahire}@i3s.unice.fr

***Abstract*—A factor slowing down the use of feature models is that either the concepts or the relations expressed in a feature model are not defined at all, or defined in an unsatisfactory manner; feature models are sometimes too vague to be analyzed by a reasoning tool. It is thus difficult to determine if the features in a feature model are arranged and structured consistently with domain knowledge and if they are accurately expressed, organized and represented. Ontology modeling can improve feature modeling by providing additional information relevant for the domain in which a feature model is constructed. Finding synergies between feature models and ontologies will aid an SPL engineer in accurately expressing, organizing and representing features in their feature models. In this paper, we look at potential benefits in using the two modeling formalisms together, we identify issues and challenges considering the gap between the two formalisms and discuss the importance of this gap. We report on our current ideas and results.**

***Keywords*-Feature modeling; Ontology; Software Product Line; Domain engineering**

I. Introduction

Software Product Line (SPL) engineering is an efficient and cost-effective approach to produce related program variants within a domain [1], [2]. Instead of considering software applications individually, the development of related programs is planned from the beginning. SPL development starts with an analysis of the *domain* to identify commonalities and variabilities (i.e., differences) between the programs in the product line. The family's common features are put into reusable assets that are then systematically assembled for deriving individual products during application engineering.

In SPL engineering, the use of feature models is becoming commonplace for describing the relationships and dependencies of a set of features belonging to a particular domain.

Besides, another widely used technique for domain analysis is ontology modeling. An ontology formally represents the knowledge by a set of concepts within a domain and the relationships between those concepts. Given such an ontology, it is then possible to reason about the properties of that domain. Ontology languages, such as Ontology Web Language (OWL) [3], are used to build a model of the domain knowledge. The OWL formal semantics specifies how to derive logical consequences of an ontology, i.e. facts not literally present in the ontology, but entailed by the semantics. Ontologies have been attracting in several domains (e.g., artificial intelligence, the Semantic Web, software engineering). Several tools (including model-based tools, e.g., [4]) have been developed for ontology representation, ontology reasoning and ontology sharing.

Both formalisms, feature models and ontologies, allow users to represent some of the knowledge in a domain (background is given in Section II).

However, the syntax, the semantics and the intention of ontologies and feature models differ. Understanding and exploiting the relationship between these two formalisms can improve SPL engineering.

The goal of the present paper is to explore the synergies between feature models and ontologies. The expected long term result is to provide mapping mechanisms such that a user can establish and take advantage of the relationship between one or more feature models and one or more ontologies.

In this paper, we suggest and describe scenarios where the relationship between ontologies and feature models can be of interest (see Section III). Then, the gap between the two formalisms is discussed (see section IV). In section V, we report on our current ideas and results according to scenarios and issues presented. Section VI reviews related work and section VII concludes and discusses some identified open issues.

II. Background

A. Feature Models

A feature model describes the valid combinations of features (or assets) that a product can have as a member of a family. Every member of a family is thus configured by a unique combination of features. The validity of a configuration is determined by the semantics of feature models and prevents the derivation of illegal configurations.

A popular definition of feature is "a distinguishable characteristic of a concept (e.g., system, component, and so on) that is relevant to some stakeholder of the concept" [5].

In our work, we have chosen to focus on *basic feature models* [6], [7]. We choose basic feature models for their

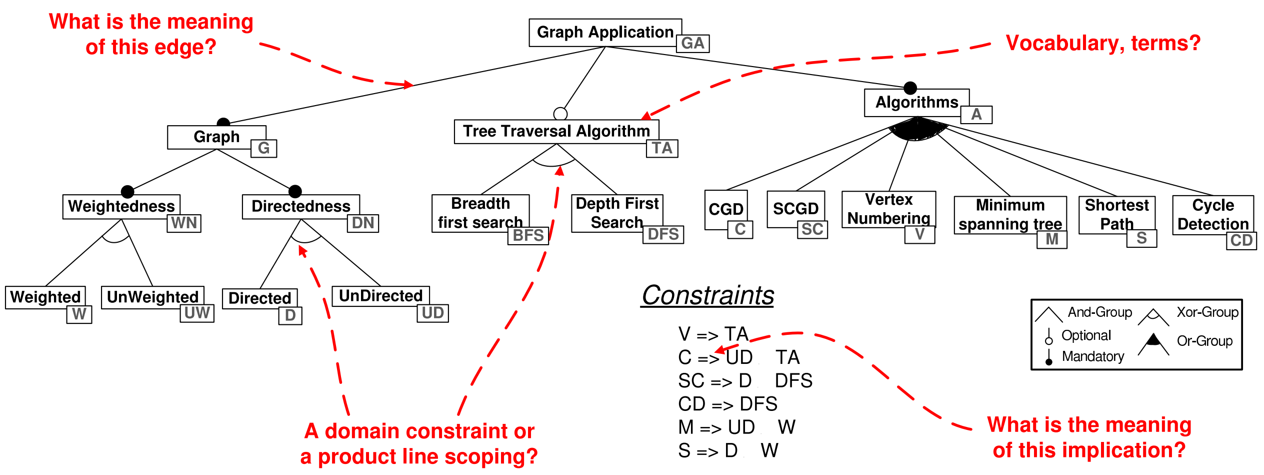


Figure 1: Feature Model of the Graph Product Line

simplicity and intuitiveness [6], their well-defined semantics [8], [7], [9], and their wide use in the SPL community.

Figure 1 is a feature model of an SPL of graph products (a popular example in this context, used in for example [10]). The graph domain is the running example used in this paper. We will explain our results on a feature model and an ontology for the graph domain.

Variability in a basic feature model is expressed through mechanisms, such as

- mandatory features: e.g., Graph is a mandatory sub-feature of the root feature Graph Application.
- optional features: e.g., Tree Traversal Algorithm is an optional (sub)feature of the root feature Graph Application.
- group constraints: (e.g. Xor- and Or- groups): e.g., Directedness has two sub-features Directed and Undirected which form an Xor-group (features are mutually exclusive)
- propositional constrains: in addition to the hierarchy of features, propositional formulas can be written by the user to constrain the selection of features. For example, Vertex Numbering implies the selection of feature Tree Traversal Algorithm.

Basic feature models are equivalent to propositional logic formula where each feature corresponds to a Boolean variable [9], [7].

B. *Ontologies*

A common definition of "Ontology" is a specification of a conceptualization [11]. We understand ontologies as digital knowledge representation. Domain knowledge include concepts, objects, and other entities that exists in an area of interest. It also includes the relationships that exist among them. Both formally and informally, an ontology can represent the knowledge in a domain.

OWL is the de-facto ontology language for the Semantic Web. It consists of three increasingly expressive sub-languages: OWL Lite, OWL DL and OWL Full. We chose OWL DL (Description Logics) since the ontology formalism is decidable, sufficiently expressive and widely used.

Description logics is a formalism of logic for representing information about classes, individuals and their relationships. Core inference problems, namely concept subsumption, consistency and instantiation, can be analyzed automatically.

In OWL DL, domain knowledge is organized as *classes* in hierarchies; instances or objects of the classes are called *individuals*; classes and individuals can be related by *properties*, (e.g., the property connected_to relates Edge to Node).

As a part of our running example, we have an ontology for the graph domain. This ontology is presented in three parts, the class hierarchy, properties and the relations among the classes. Figure 2a shows the class hierarchy, Figure 2b shows the properties and Figure 3 shows the relations. Together these three figures are the complete ontology.

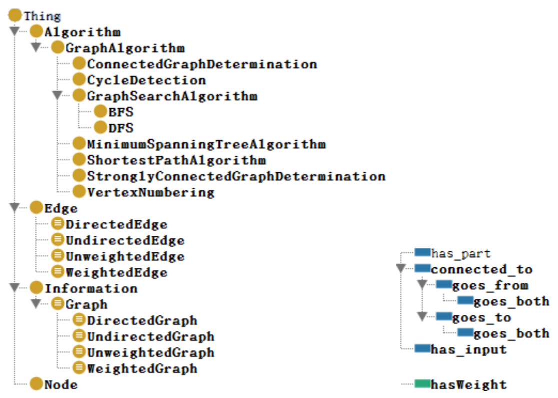
The relations are specified in description logic syntax as the relations were modeled in the Protégé tool for ontologies. An explanation of description logic syntax is not presented here due to space constrains, but the syntax is common for description logic and the domain of the ontology is well known.

III. Possible Enhancements of SPLE using Ontologies

A. *How Feature Models Lack Some Interesting Information*

Let us take a closer look at the feature model in Figure 1 to see how it is lacking some interesting information.

- What is the meaning of the edge decorated with a filled black circle that relates Graph Application and



(a) Class hierarchy (b) Property hierarchy

Figure 2:

$$
\begin{aligned}
&Graph \equiv (\forall has\_part.(Edge \sqcup Node)) \sqcap (\geq 1\ has\_part.Node)\\
&DirectedGraph \equiv Graph \sqcap \forall has\_part.DirectedEdge\\
&UndirectedGraph \equiv Graph \sqcap \forall has\_part.UndirectedEdge\\
&UnweightedGraph \equiv Graph \sqcap \forall has\_part.UnweightedEdge\\
&WeightedGraph \equiv Graph \sqcap \forall has\_part.WeightedEdge
\end{aligned}
$$

$$
\begin{aligned}
&Edge \sqsubseteq (= 2\ connected\_to.Node)\\
&Edge \sqsubseteq (\leq 1\ hasWeight.integer)\\
&DirectedEdge \equiv Edge \sqcap (= 1\ goes\_from.Node)\\
&\qquad \sqcap (= 1\ goes\_to.Node)\\
&UndirectedEdge \equiv Edge \sqcap (= 2\ goes\_both.Node)\\
&UnweightedEdge \equiv Edge \sqcap (= 0\ hasWeight.integer)\\
&WeightedEdge \equiv Edge \sqcap (= 1\ hasWeight.integer)
\end{aligned}
$$

$$
\begin{aligned}
&Algorithm \sqsubseteq \exists has\_input.Information\\
&GraphAlgorithm \sqsubseteq \exists has\_input.Graph\\
&ConnectedGraphDetermination \sqsubseteq\\
&\qquad (= 1\ has\_part.GraphSearchAlgorithm)\\
&ConnectedGraphDetermination \sqsubseteq\\
&\qquad (= 1\ has\_input.UndirectedGraph)\\
&CycleDetection \sqsubseteq (= 1\ has\_part.DFS)\\
&MinimumSpanningTreeAlgorithm\\
&\qquad \sqsubseteq (= 1\ has\_input.(UndirectedGraph \sqcap WeightedGraph))\\
&ShortestPathAlgorithm \sqsubseteq\\
&\qquad (= 1\ has\_input(DirectedGraph \sqcap WeightedGraph))\\
&StronglyConnectedGraphDetermination \sqsubseteq (= 1\ has\_part.DFS)\\
&StronglyConnectedGraphDetermination \sqsubseteq\\
&\qquad (= 1\ has\_input.DirectedGraph)\\
&VertexNumbering \sqsubseteq (= 1\ has\_part.GraphSearchAlgorithm)
\end{aligned}
$$

$$
\begin{aligned}
&\top \sqsubseteq \forall connectedTo.Edge\\
&\top \sqsubseteq \forall connectedTo^{-}.Node\\
&\top \sqsubseteq \forall has\_input.Algorithm\\
&\top \sqsubseteq \forall has\_input^{-}.Information
\end{aligned}
$$

Figure 3: Relations of the graph ontology in DL syntax

Graph? Edges in feature models are parent-child relations between features. They imply that the parent must be selected before the child is selected. The relationship between Graph Application and Graph can be an aggregation relationship, meaning, for example, that a graph application "owns", "treats", or "hasInputs" graphs or a generalization relationship, meaning that a graph application "isA", "refines" or "inheritsFrom" graphs. Hence, the reason for a relationships between features in the hierarchy is explained in a lower level of detail than what an ontology can.

- What is the relationship between the features Vertex Numbering and Tree Traversal Algorithm? The propositional formula $V \Rightarrow TA$ expresses clearly a relationship between those two features. However, the detail of the reason behind the constraint is limited to a Boolean implication on its inclusion. It does not say much about the nature of feature dependency. Once again, the implication can be interpreted as a traceability link (e.g., "implementedBy"), a software dependency (e.g. "requiredBy" or "shouldInclude"). Hence, also here an ontology supports a higher level of detail.
- Are constraints (e.g. Group-constraints) in the feature model consistent with the knowledge of the domain? For instance, the inclusion of features Directed and Undirected are mutually exclusive. This is reasonable as the edges in a graph cannot be both directed and undirected at the same time, something that can be expressed in an ontology.
- Is the vocabulary standard and semantically accurate for the domain? For example, are the terms used in the feature model proper for the graph domain? In an ontology of the graph domain, we find terms and their relationships which are, as is intended in an ontology, accurate.

B. Using Ontologies to Improve SPL engineering

In our opinion, a factor slowing down the use of feature models is that either the concepts or the relations expressed in a feature model are not defined at all, or defined in an unsatisfactory manner. Feature models are sometimes too vague to be analyzed by a reasoning tool. It is thus difficult to determine if the domain knowledge is accurately expressed, organized and represented within a feature model.

An ontology can improve the use of feature models in the following ways:

- define the semantics of features and their relationships
- provide accurate vocabulary and terms from a domain to SPL engineers

If we established a link between a feature model and an ontology from its domain, several possibilities emerge which we will discuss in the following subsections. We will use the running example with a graph product line, described in

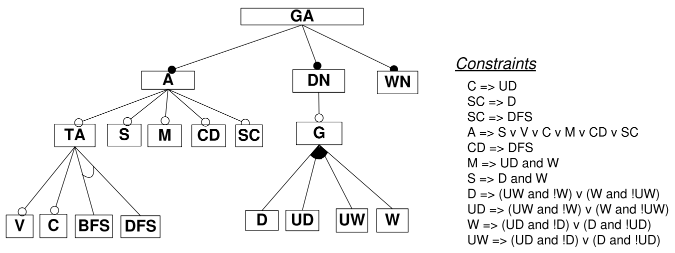

GA
A
DN
WN
TA
S
M
CD
SC
G
V
C
BFS
DFS
D
UD
UW
W
Constraints
C => UD
SC => D
SC => DFS
A => S v V v C v M v CD v SC
CD => DFS
M => UD and W
S => D and W
D => (UW and !W) v (W and !UW)
UD => (UW and !W) v (W and !UW)
W => (UD and !D) v (D and !UD)
UW => (UD and !D) v (D and !UD)

Figure 4: A feature model equivalent to feature model of Figure 1

Figure 1 and an ontology of the graph domain in figures 2 and 3.

1) Consistency checking between a feature model and an ontology: The first step is to establish a mapping between the feature model and the ontology. In particular, some features of the feature model should refer to concepts expressed in the ontology. For example, the feature Graph can refer to the class Graph; the edge which relates Algorithm and Shortest Path can be mapped to a subclass relation in the ontology between Algorithm and ShortestPathAlgorithm.

Once the mapping is established, it is possible to state if the feature model is consistent with the ontology. Are the domain constraints expressed in the ontology expressed in the feature model? Are there missing or too strong constraints in the feature model?

2) Refactoring the hierarchy of a feature model[1]*:* Let us consider the feature model in Figure 4. At first sight, this feature model seems to be really different from the feature model of Figure 1. However, the two feature models represent exactly the same set of valid configurations. The only difference is the way features are organized in the feature model, that is, its hierarchy. An ontology may assist a user in refactoring the hierarchy and in better organizing the relations between features. For example, there is a direct relation between the feature Tree Traversal Algorithm and its sub-feature Vertex Numbering in the feature model of Figure 4 whereas, in the ontology, there is no relation between the class GraphSearchAlgorithm and the class VertexNumbering.

3) Explaining the hierarchy of a feature model: Using an ontology, the semantics of the relationships between features may be automatically inferred and, thus, made explicit to users.

For example, we can deduce from the ontology that there is a sub-sumption relation between features Breadth First Search (resp. Depth First Search) and Tree Traversal Algorithm. Similarly, we can explain why the features Breadth First Search and Depth First Search are mutually exclusive: the corresponding classes in the ontology are disjoint.

IV. Understanding the Gap Between the Two Formalisms

During our research, we have pin pointed several differences between feature models and ontologies as formalisms. The differences presented are related to specifying the relationship between two independently made models: one feature model and one ontology from the same domain, such as those in our running example. It is important to understand this gap in order to find synergies between them.

A. The Open World and The Closed World Assumptions

An important difference between the two formalisms is the fact that one uses the open world assumption and the other uses the closed world assumption. The feature model is oriented towards system construction so that the represented knowledge is implicitly viewed as being complete.

OWL, in contrast, interprets models as potentially representing partial knowledge. An example should make the concerns clear. Let us consider that we have one edge between two features in a feature model, one relation between two classes in an ontology and the classes are mapped to the features. We cannot necessarily infer that the edge of the feature model maps to the relation of the ontology. The reason is that the open world assumption states that we cannot rule out the existence of another relation in the ontology. We do not know whether an unstated relation is the one closest to the intended meaning of the edge in the feature model.

For example, consider the implication between Shortest Path and Directed. In the ontology, there is a relation between these as has_input, but the relation could, if we consider the graph domain, be a relation on has_domain or $is\_defined\_for$.

B. The Models' Purposes

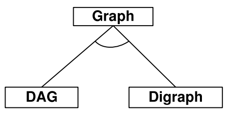
A related issue is found in the nature of the two models. A feature model is by nature focused on modeling something specific, a family of products within a domain, while an ontology is focused on modeling something general, the knowledge in the domain.

For example, one should not say a feature model is inconsistent with an ontology if it is more restricted than the ontology: an algorithm implemented for an SPL might be for more restricted graphs than what is know to be possible within the domain. Hence, one should regard something which is more strict in the feature model as consistent with something less strict in the ontology.

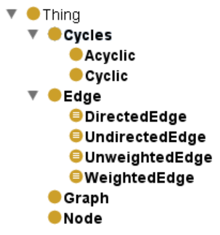
C. Granularity

If we have two models that are dedicated to the representation of the same things, we might still have to handle the

[1]We rely on the terminology used in [12]. Let f and g be FMs, and let $[\![f]\!]$ (resp. $[\![g]\!]$) denote the set of configurations for g. f is a *refactoring* of g if $[\![g]\!] = [\![f]\!]$, i.e., f and g represent the same set of configurations.



(a) Lower granularity



(b) Higher granularity

Figure 5: Example of different granularity

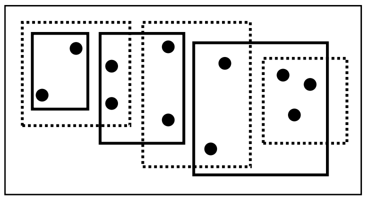


Figure 6: Abstract example of the different granularities and focuses

mismatch between the granularity and the focus of the two models, so let us look at these two things.

Figure 5 illustrates a difference in information granularity of two models, a feature model and an ontology. Sub-figure 5a models the variability of a graph in an SPL. In this feature model, you are allowed to select one of two different graph types, a DAG (directed acyclic graph) or a digraph (directed graph). Each of these graphs has some sub-properties. Sub-figure 5b is an ontology of a graph with finer granularity (only the class hierarchy is depicted). In this model, the different properties of a graph are represented. Even though both the feature model and the ontolog model the same thing, they do so at different levels of detail. They are said to have different granularity. In order to relate a feature model and an ontology, granularity differences must first be resolved.

D. Focus

Figure 6 shows the impact of having different focus in an ontology and in a feature model. The dots symbolize classes at a high level of granularity. The squares represents two different coarse classifications. Let us say that the first type of square represents information of a feature model while the second type of square represents information of an ontology. If the dots are different utilities such as, let say, mobile phones, cameras or mp3-players then a mobile phone may be classified as a camera in one model and a mp3-player in another since it has all three capabilities. The dots compared to the two types of squares are at a finer level of granularity. The two types of squares are at the same granularity but have different focus. As a result, the two models we want to compare must, in addition to the same granularity, also have the same focus.

E. Model Information Overlap

In a model, we have information about the things we want to model. Specifying and delimiting the information content that overlaps is important. In the relation between the two models, we only want to relate the things that talk about the same things. Identifying overlapping parts cannot be fully automated, and it requires the users' knowledge.

For example, in the running example, the feature model has information about a graph application. This information is not found in the ontology. Hence, this information is not a part of the overlapping information of the two models, and must hence be excluded when performing a consistency check, for example.

F. Cardinalities

Feature models have different capabilities than ontologies when it comes to cardinalities. For example, take out running example. We use a feature model to model different kinds of graphs which this application can work on. After having specified one type of graph using this feature model, we find ourselves unable to model another graph. What if the application should support two types of graphs? The reason is that a feature model only allows us to either have the graph feature or not; we cannot decide ourselves how many graphs we would like to specify. This is a result of the fact that feature models have a zero or one cardinality for each feature, while an ontology allows us to have a zero-to-many cardinality.

Feature models with cardinalities and cloning mechanisms have been investigated in [13]. Such an addition brings feature models closer to ontologies [6]. Also, instantiating a feature model several times solves this problem, but this is also an expansion of basic feature models which is the context of this paper.

V. Current Ideas

We will now look at some ideas around the synergies between feature models and ontologies. We continue using our running example. In Figure 7, the following scenario is considered: a user wants to reason about the relationship between an existing feature mode, $Feature\ Model_e$, representing the feature model of the graph product line, and an existing ontology, $Ontology_e$, representing the ontology of the graph domain.

A. Strategies Considered

The semantics of $Feature\ Model_e$ and $Ontology_e$ are two different formalisms. It might be easier to reason on the models in a common representation than it is to directly reason about the two models (①).

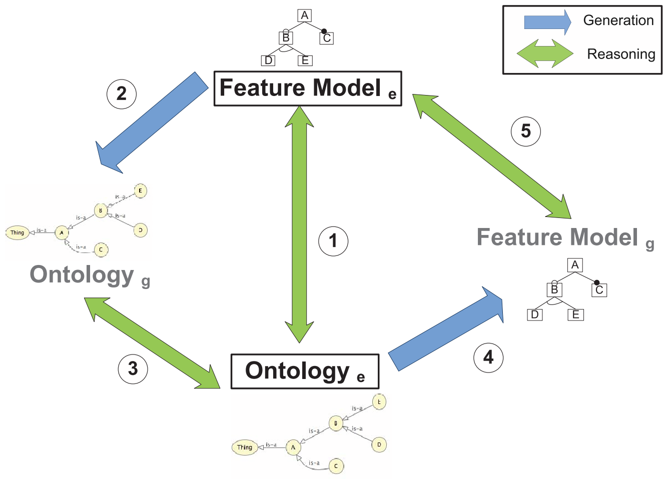


Figure 7: Strategies to reason about a feature model and an ontology

From Feature Model to Ontology: The first strategy is to generate an ontology, called $Ontology_g$ from $Feature\ Model_e$ (②). Wang et al. [14], [15] propose to transform a basic feature models to an OWL ontology. We can use the proposed transformation of Wang et al. to OWL to generate the ontology $Ontology_e$ from $Feature\ Model_e$.

If we manage to convert it, then we can work on the relation between them in this new form (③). One benefit is to take advantage of the reasoning capabilities on the OWL formalism to reason about feature models. The drawback is that comparing two ontologies is far from being trivial to achieve (e.g., see [16] for an overall view of the *alignment* problem).

From Ontology to Feature Model: A second strategy is to generate a feature model $Feature\ Model_g$ from an $Ontology_e$ (④) and then to consider $Feature\ Model_e$ and $Feature\ Model_g$ (⑤). In this case, existing research, techniques and tool support on computing properties of or relationships among feature models [12], [17], [18] may be reused.

B. Mapping Rules

The following are some thoughts on possible mapping rules between feature models and ontologies. The mapping rules are all bidirectional. They can be utilized for reasoning (①), for performing generations (② and ④) as seen in Figure 7.

feature model $\leftrightarrow$ class: A feature model as a whole maps to a single class in the ontology.

There are several arguments that justify this mapping. (1) The purpose of feature models is to model the variability of one SPL. The name of the SPL is usually the name of the root node, and thus of the feature model. (2) Cross tree constraints in the feature model live within the feature model globally; in other words, they belong to the root node. Their place in the ontology is the class that maps the feature model. (3) The structure of a configuration is in an ontology one class, representing the root node with the sub-features as relations in it. This is similar to the encoding proposed in [15].

feature $\leftrightarrow$ class: Features can be mapped to partial and complete classes by name equality or user specification.

A class in an ontology can take many forms. There are partial and complete classes. Partial classes are not fully defined; hence, they must be named. They specify necessary super classes but do not have a complete definition. Complete classes are fully defined; hence, need not be named.

group $\leftrightarrow$ class: Groups in feature models can be And-, Xor- or Or-groups and establish a relation between one feature and its sub features. A feature with a child group sometimes names the group. A typical example is a super class and its sub-classes with an inheritance relationship. In those cases the group is taken to be the feature and hence follows the same mapping rules as features. If the feature does not name the group then the group is anonymous. In those cases it can either be mapped to a class by inference or by user specification.

propositional logic $\leftrightarrow$ description logic: In [15], the mapping of *requires* and *excludes* is given. This part of the mapping can be extended by taking advantage of existing research on the encoding of DL as a SAT problem [19].

name	PL	DL
and	$A \wedge B$	$hasA \sqcap hasB$
or	$A \vee B$	$hasA \sqcup hasB$
not	$\neg A$	$\neg hasA$

Table I: PL to DL mappings

Table I shows some possible mappings between the two logical formalisms. In the PL (Propositional Logic) column we see constraints typically found in feature diagrams. In the DL (Description Logic) column we find the corresponding mapping as used in [15].

C. Model Consistency

When we manage to reason successfully about the relationship between a feature model and an ontology, it is interesting to determine whether the two models are consistent. It may be tempting to define the consistency of a feature model and an ontology in terms of configurations and individuals. There are several arguments against this. (1) A feature model has instances (configurations), but an ontology does not. An ontology contains instances of classes, namely individuals. (2) If we tried to compare configurations with individuals of a class, we would run into the next problem which is the open world assumption. Each class has an infinite amount of possible individuals. One can close a class meaning that it has a finite number of possibilities. One cannot assume, however, that classes of an ontology are closed. Actually, it is quite uncommon to close classes in ontologies. (3) Even if one only compares configurations with individuals of a closed class, one would get into the problem of cardinalities. At this point one has made too many assumptions to make this approach feasible. A consistency checker must cope with most the model differences presented in section IV. A definition which copes with some of these is: *two models are consistent if the information in one does not contradict the information in the other*. This definition deals with the problem of partial information content overlap by only considering the intersection of information. The open world assumption is handled by understanding the information content in the ontology while taking into account the open world assumption for the ontology when understanding the meaning of the ontology model upon checking if it contradicts with the feature model. Granularity and focus are still problems, as the models must be understood in common terms before the consistency can be evaluated.

VI. Related Work

Czarnecki et al. investigate the relation between feature models and rich modeling languages such as ontologies. They only consider class modeling-based ontologies in their work [6].

The authors defend the idea that feature models are views on ontologies in the sense that a feature model establishes a scope and configures ontologies. In [6], research directions are suggested for the combined use of feature models and ontologies. *View projection* refers to the process of having feature models obtained automatically or semi-automatically from an existing ontology. Another option, called view integration, is to design feature models more or less independently with some ontology in mind and then align the views.

Our work is mainly inspired by these research suggestions and investigates further in these directions.

The two proposed views can apply to OWL-based ontologies. The view projection corresponds to the scenario where a feature model is generated from an ontology (see section V). Consistency checking between a feature model and an ontology can be part of the view integration.

In [20], a bidirectional mapping between class/object models and feature models/configurations is presented (with tool support). The class models considered are Ecore class models while the feature models formalism uses more complex cardinalities. Ecore class models can be considered a simple ontology. The authors argue that both class modeling and cardinality-based feature modeling have similar expressive power. In our work, the gap considered between the two formalisms – basic feature models and OWL-based ontology – is much coarser.

Wang et al. propose to reduce basic feature models to OWL language [14], [15]. The primary motivation behind the proposed encoding is to use the ontology reasoning framework to perform automated analysis over the OWL representation of the feature models. For instance, OWL reasoning engines can check the consistency of a feature model or the validity of a configuration. As a result, their work can be considered as yet another extension of the research on automated analysis of feature models; namely computing properties of feature models. In the same way, feature models can be encoded to propositional formulas or to constraint satisfaction problems, for which off-the-shelf tools - SAT solvers, BDD tool [18], CSP solvers [21] - can validate properties of models [17].

The authors do not consider scenarios where feature models are related to existing ontologies. The reduction of feature models to an ontology can have other interests and applications as proposed in this paper (e.g., consistency checking between the two models) .

In [22], feature models are enriched with ontologies. The approach consists in annotating some features of a feature model with information (e.g. Non functional properties) contained in an existing ontology. Once a feature model is fully annotated with an ontology, analysis and reasoning is achieved in the OWL logical space. To fulfill this purpose, the initial feature model is represented in OWL language (as proposed in Wang et al. [14], [15]) which enables them to take advantages of ontology annotation capabilities.

In [23], the authors carry on their approach and propose an algorithm to specialize feature models annotated with ontologies. Features which do not satisfy the ontology annotations are discarded pruning the feature model into a new feature model whose set of possible configurations is a subset of the original feature model, a specialization [13].

Bachmeyer and Delugach consider conceptual graphs to represent feature models [24]. The purpose of their work is to produce feature models that "have a more natural, and more easily expressed mapping to the problem domain". An automatic mapping is proposed such that a feature model can be expressed in the conceptual graph formalism.

Asikainen et al. [25] introduce Forfamel, a conceptual foundation for feature modeling. Forfamel includes a few additional constructs and concepts (such as attributes, types, instances, etc.) which provides an expressiveness closer to rich modeling languages and ontology.

VII. Conclusion and Perspectives

Since their introduction by Kang et al. [26], feature models have evolved in several aspects, including their expressiveness, semantics and usage in SPL engineering. We think ontologies can play an important role in feature modeling. The present paper explored the synergies between the two formalisms.

Several scenarios where ontologies can be applied to feature modeling have been considered. We also identify several issues in the gap between the two formalisms. Our first experiments focus on the basic feature models formalism. We investigated techniques which relates a feature model with an existing ontology and looked at the consistency of their relationship.

The issues listed below remain to be resolved and are opened to research:

I-1: Assisting the user during the manual extraction of a feature model from an ontology. Tools can propose an extraction which acts as a preliminary step before the user intervention. A tool can also provide to the user some suggestions (e.g., which properties of the ontology are of interest) or interactively prevent inconsistent choices.

I-2: Determining the accuracy of information content. As ontologies are more powerful than feature models, some information of an ontology cannot be represented in the feature model in the generation step. From a user perspective, we have to determine how the gap between basic feature models and OWL DL ontologies can be practically handled. Lessons learned can motivate the studying of richer feature model formalism. Another strategy considered in the paper is to generate an ontology from a feature model. In this case, using the expressiveness of ontologies can leverage the reasoning about feature models.

I-3: Threats to the relationship between feature models and ontologies are: *i)* the amount of user's effort can turn out to be very important considering the benefits achieved ; *ii)* the feature models' users are not necessary experts or even familiar with ontologies. Assessment with case studies can help to improve the understanding of the relationship between feature models and ontologies and its practical interest. We are already using ontologies and feature models in the medical imaging domain [27], [28], and we plan to do the same in the video surveillance domain [29].

Acknowledgments

The work presented here has been developed within the VERDE project ITEA 2 - ip8020. VERDE is a project within the ITEA 2 - Eureka framework.

References

[1] P. Clements and L. M. Northrop, *Software Product Lines : Practices and Patterns*. Addison-Wesley Professional, 2001.

[2] K. Pohl, G. Böckle, and F. J. van der Linden, *Software Product Line Engineering: Foundations, Principles and Techniques*. Springer, 2005.

[3] M. Dean and G. Schreiber, "OWL web ontology language reference," W3C, W3C Recommendation, February 2004.

[4] F. Silva Parreiras and S. Staab, "Using Ontologies with UML Class-based Modeling: The TwoUse Approach." *Data and Knowledge Engineering*, 2010.

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

[6] K. Czarnecki, C. H. Peter Kim, and K. T. Kalleberg, "Feature models are views on ontologies," in *SPLC '06*. IEEE, 2006, pp. 41–51.

[7] K. Czarnecki and A. Wasowski, "Feature diagrams and logics: There and back again," in *SPLC'07*. IEEE, 2007, pp. 23–34.

[8] P.-Y. Schobbens, P. Heymans, J.-C. Trigaux, and Y. Bontemps, "Generic semantics of feature diagrams," *Comput. Netw.*, vol. 51, no. 2, pp. 456–479, 2007.

[9] D. Batory, "Feature models, grammars, and propositional formulas," in *SPLC'05*. Springer, 2005, pp. 7–20.

[10] R. E. Lopez-Herrejon and D. S. Batory, "A standard problem for evaluating product-line methodologies," in *GCSE '01*. Springer-Verlag, 2001, pp. 10–24.

[11] T. R. Gruber, "Towards principles for the design of ontologies used for knowledge sharing," in *Formal Ontology in Conceptual Analysis and Knowledge Representation*, N. Guarino and R. Poli, Eds., 1993.

[12] T. Thüm, D. Batory, and C. Kästner, "Reasoning about edits to feature models," in *ICSE'09*. IEEE, 2009.

[13] K. Czarnecki, S. Helsen, and U. Eisenecker, "Staged Configuration through Specialization and Multilevel Configuration of Feature Models," *Software Process: Improvement and Practice*, vol. 10, no. 2, pp. 143–169, 2005.

[14] H. Wang, Y. F. Li, J. Sun, and H. A. Zhang, "A semantic web approach to feature modeling and verification." in *Workshop on Semantic Web Enabled Software Engineering (SWESE'05)*, November 2005.

[15] H. H. Wang, Y. F. Li, J. Sun, H. Zhang, and J. Pan, "Verifying feature models using owl," *Web Semant.*, vol. 5, no. 2, pp. 117–129, 2007.

[16] J. Euzenat and P. Shvaiko, *Ontology matching*. Springer, 2007.

[17] D. Batory, D. Benavides, and A. Ruiz-Cortés, "Automated analysis of feature models: Challenges ahead," *Communications of the ACM*, vol. December, 2006.

[18] M. Mendonca, A. Wasowski, and K. Czarnecki, "Sat-based analysis of feature models is easy," in *SPLC '09*. IEEE, 2009, pp. 231–240.

[19] F. Gasse and V. Haarslev, "Expressive description logics via sat: The story so far," in *7th International Workshop on Satisfiability Modulo Theories (SMT '09)*, 2009.

[20] M. Stephan and M. Antkiewicz, "Ecore.fmp, a tool for editing and instantiating class models as feature models," University of Waterloo, Tech. Rep. 2008-08, 2008.

[21] D. Benavides, A. Ruiz-Cortés, and P. Trinidad, "Automated reasoning on feature models," *CAiSE '05*, vol. LNCS 3520, pp. 491–503, 2005.

[22] N. Kaviani, B. Mohabbati, D. Gasevic, and M. Finke, "Semantic annotations of feature models for dynamic product configuration in ubiquitous environments," in *4th International Workshop on Semantic Web Enabled Software Engineering*. at 7th International Semantic Web Conference, 2008.

[23] N. Kaviani, B. Mohabbati, and D. Gasevic, "Semantic variability modeling for multi-staged service composition," in *Service-Oriented Architectures and Software Product Lines (SOAPL '09)*, workshop of SPLC '09. IEEE, 2009.

[24] R. Bachmeyer and H. Delugach, "A conceptual graph approach to feature modeling," *Conceptual Structures: Knowledge Architectures for Smart Applications*, pp. 179–191, 2007.

[25] T. Asikainen, T. Mannisto, and T. Soininen, "A unified conceptual foundation for feature modelling," in *SPLC '06*. IEEE, 2006, pp. 31–40.

[26] K. Kang, S. Cohen, J. Hess, W. Novak, and S. Peterson, "Feature-Oriented Domain Analysis (FODA) Feasibility Study," Software Engineering Institute, Tech. Rep. CMU/SEI-90-TR-21, Nov. 1990.

[27] M. Acher, P. Collet, P. Lahire, and J. Montagnat, "Imaging Services on the Grid as a Product Line: Requirements and Architecture," in *Service-Oriented Architectures and Software Product Lines (SOAPL '08)*, workshop of SPLC '08. IEEE, 2008.

[28] M. Acher, P. Collet, P. Lahire, and R. France, "Managing Variability in Workflow with Feature Model Composition Operators," in *9th International Conference on Software Composition (SC'10)*, ser. Software Composition, vol. LNCS. Springer, Jun. 2010, p. 16.

[29] M. Acher, P. Collet, F. Fleurey, P. Lahire, S. Moisan, and J.-P. Rigault, "Modeling Context and Dynamic Adaptations with Feature Models," in *4th International Workshop Models@run.time at Models '09 (MRT' 09)*, 2009, p. 10.