IDENTIFYING DEAD FEATURES AND THEIR CAUSES IN PRODUCT LINE MODELS: AN ONTOLOGICAL APPROACH

IDENTIFICANDO CARACTERÍSTICAS MUERTAS Y SUS CAUSAS EN MODELOS DE LÍNEAS DE PRODUCTOS: UN ENFOQUE ONTOLÓGICO

GLORIA-LUCIA GIRALDO
PhD, Associate Professor, Universidad Nacional de Colombia Sede Medellín, Colombia, glgiraldog@unal.edu.co

LUISA RINCÓN-PEREZ
Master of Engineering- Engineering Systems, Universidad Nacional de Colombia Sede Medellín, Colombia, lufrinconpe@unal.edu.co

RAUL MAZO
PhD, Associate Professor, CRI, Université Paris 1 Panthéon Sorbonne, raul.mazo@univ-paris1.fr

ABSTRACT: Feature Models (FMs) are a notation to represent differences and commonalities between products derived from a product line. However, product line modelers could unintentionally incorporate dead features in FMs. A dead feature is a type of defect, which implies that one or more features are not present in any product of the product line. Some authors have used ontologies in product lines, but they have not exploited ontology reasoning to identify and explain causes for defects in FMs in natural language. In this paper, we propose an ontology that represents FMs in OWL (Web Ontology Language). Then, we use SQWRL (Semantic Query-enhanced Web Rule Language) to identify dead features in a FM and identify and explain certain causes of this defect in natural language. Our preliminary empirical evaluation confirms the benefits of our approach.

Key words: Product lines, feature models, ontologies, dead features, SQWRL.

RESUMEN: Los modelos de características (en inglés *Feature Models FMs*) son una notación para representar diferencias y similitudes entre productos derivados de una línea de productos. Sin embargo, quienes modelan la línea de productos pueden introducir sin intención en los *FMs* defectos como las características muertas. Una característica es muerta si no puede estar presente en ningún producto derivado de la línea de productos. Algunos autores han identificado características muertas en los FMs, pero ninguno ha aprovechado las capacidades de razonamiento de las ontologías para identificar y explicar las causas de estos defectos en lenguaje natural. En este trabajo, se propone una ontología para identificar las características muertas en un FM y se proponen consultas sobre la ontología, para identificar y explicar en lenguaje natural ciertas causas de las características muertas detectadas. Nuestra evaluación empírica preliminar confirma los beneficios de nuestra propuesta.

Palabras clave: Líneas de productos, modelo de características, ontologías, características muertas, SQWRL.

1. INTRODUCTION

A product line is a family of related products distinguished by different sets of features that each product provides [1]. A particular application of product line is the software product line (SPL). Software Product Line Engineering (SPLE) is thus the software development paradigm geared for the construction of SPLs. Extensive research and industrial experience widely prove the significant benefits of SPLE practices, which among them are reduced time to market, increased asset reuse and increased software quality [2]. In order to do that, SPLE usually uses Product Line Models (PLMs) to represent the correct combination of features that represent valid products.

A common notation to represent PLMs is Feature Models (FMs). FMs describe the features, their relations and the valid feature combinations of a product line [3]. FMs have also proven useful to communicate with customers and other stakeholders, such as marketing representatives, managers, production engineers, system architects, etc. Consequently, having FMs that correctly represent the domain of the product line is of paramount importance for ensuring quality in SPLE. However, creating feature models with features that correctly represent the domain described by the model is not trivial [4]. In fact, when a FM is constructed, defects may be unintentionally introduced, which decreases the quality of the FM and hence also the

expected benefits of product line. Dead features are one such defect. A feature is dead if it cannot appear in any products of the product line [5–8].

Some studies in the literature automatically identify whether a FM present dead features or not [5, 9–11]. However, few studies have focused on identifying causes for such defect [12, 13]. Identifying the cause consist in identifying the dependencies that, combined in a certain manner, produce a dead feature. Such identification helps product line engineers to understand the problem and to determine the best solution to fix dead features [4, 14]. In addition, in an end-users configuration process, it is important to identify defects and explain the cause of these defects to users [15].

FMs and ontologies are comparable because both represent concepts of a particular domain and their dependencies [16]. However, FMs only offer a graphical means to represent a particular domain, whereas ontologies also offer an efficient mechanism to reason on domain models.

In this paper, we discuss an approach based on OWL-DL (*Web Ontology Language–Description Logic*) [17] ontology and SQWRL (*Semantic Query-enhanced Web Rule Language*) [18]. The ontologies are formal domain models, which have powerful inference mechanisms. The ontologies are recommended for sharing terminologies and understanding. Therefore, modeling with ontologies offers interoperability, reusability and extensibility. We represent our ontology in OWL-DL because this formalism provides computational completeness and expressiveness for representing knowledge [19]. Moreover, we use a rule-based language because the rules are a natural and declarative way to represent knowledge [20]. SQWRL is a rule-based language to extract information from OWL ontologies. Therefore, we use this language to identify dead features in FMs, and some of their causes from the proposed ontology.

Our proposal has two main contributions. First, it provides an ontology that represents FM concepts; second, it identifies and explains in natural language certain causes that produce dead features.

The remainder of the paper is as follows. In Section 2, we give a brief overview of the necessary concepts for understanding the study described herein. Section 3 presents our approach to identify the causes for dead features in FMs. In Section 4, we present the preliminary validation of our proposal. In Section 5, we present related research. Finally, Section 6 presents the conclusions and suggests future research directions.

2. GENERAL CONCEPTS

Feature Models

Feature Models (FMs) are a notation for representing product line models. Using this notation, a feature is a distinctive element, which is visible to users. Each feature is a node in a tree structure, and the model dependencies are arcs [3].

The tree structure represents hierarchical organization of the features, wherein only one feature is the model root feature. In addition, except for the root, each feature has a parent feature [3]. Figure 2 shows a FM, which exemplifies the application of our proposal. Features can have different types of dependencies. Table 1 describes and graphically represents each type of dependency.
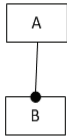
After Kang et al. [3] reported a first notation of FMs, other authors proposed extensions to the original notation [21] (e.g., the group cardinality dependency [22]).

Figure 1 shows an UML-based meta-model for a cardinality base FM. This meta-model relates the concepts presented in Table 1.

of *Utility Functions*, *Settings,* and *Media* features, among others. As shows Figure 2, each child feature in the *Media* feature is optional. Additionally, each child feature in the *Utility Functions* and *Settings* features are mandatory. Each dependency connects two features with a unique nomenclature for easy identification.

In order to illustrate our approach, we intentionally introduced in the original model four dead features (*MSN, Camera, VGA and Megapixels*). For that purpose, we use two additional dependencies (R16 and R17).

Table 1. Types of dependencies in FMs

| Notation	Type of Dependency
	Mandatory [3] Child feature B should be included in all valid products containing the parent feature A and vice versa. It a feature is mandatory and all its ancestors are also mandatory, then, this feature is a full-mandatory feature [5].
	Optional [3] Child feature B may or may not be included in valid products containing parent feature A. However, if feature B is included in a product, its parent A should be included too.
	Group cardinality [22] Represents the minimum (m) and the maximum (n) number of child features (B...C) grouped in a cardinality (<m..n>) that a product can have when the parent feature (A) is included in the product. If at least one of the child features is included into a product, the parent feature should be included too.
Transverse Dependencies	
	Requires [3] Feature B should be included in valid products with feature A. This dependency is unidirectional.
	Excludes [3] Features A and B cannot be in valid products at the same time. This dependency is bidirectional.
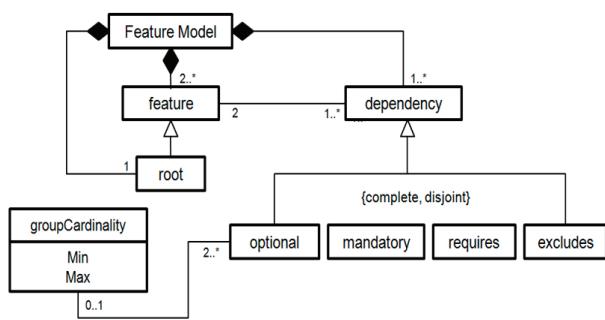
Figure 1. FM meta-model based on the one proposed by Mazo *et al.* [15]

2.2. Application Example

Figure 2 shows a reduced version of a FM based on the one proposed by Segura for mobile phones [23]. In this example, a *Mobile Phone* is composed
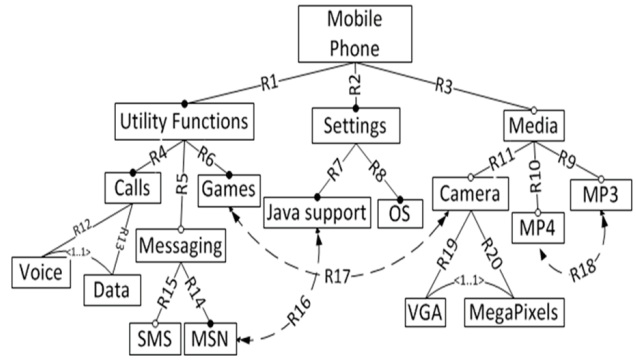
Figure 2. Summary of a mobile phone FM based on Segura's proposal [23]

2.3. Dead features

Features are distinctive elements that are visible to user [3]. A feature is dead when it is not present in any product of the product line [5–8]. When a FM has dead features, the model is not an accurate representation of the domain. Indeed, if a feature belongs to a FM, the feature is important for the domain that we want to represent. Therefore, it should be possible to incorporate that feature in at least one product of the product line [7].

2.4. Ontologies

An ontology is a formal explicit specification for a shared conceptualization [24, 25]. In the same way that FMs, ontologies help to identify and define the domain basic concepts and the dependencies among them.

Representing information with ontologies aids the identification and definition of the basic terms of a domain. In addition, ontologies represent the dependencies and rules for combining such terms, and provide a common vocabulary for the domain model. Ontologies comprises classes, instances, properties and constraints [26].

Classes are the main concepts related to the ontology domain. Instances represent objects in the domain of interest. Properties are object properties or data-type properties: Object properties relate ontology instance among them, whereas data-type properties relate ontology instances with concrete values, for example, an integer value. Finally, constraints describe the restrictions that instances must satisfy in order to belong to a class [26].

2.5. SQWRL Queries

The Semantic Query-enhanced Web Rule Language (SQWRL) is a language that provides query operations for ontologies represented in OWL [18]. A SQWRL query comprises an antecedent and a consequent expressed in terms of OWL classes and properties. The antecedent defines the criteria that instances must satisfy to be selected, and the consequent specifies the instances to select in the query results [18]. each SQWRL uses classes and properties defined in the proposed ontology to query for information of the FM represented as ontology instances. A semantic reasoner, such as JESS (Java Expert System Shell) [27], executes SQWRL queries.

3. PROPOSED SOLUTION

In the previous section, we described the basic concepts underlying our work. Following sub-sections present our approach, which uses ontologies and SQWRL to identify certain dead features in FMs, and to explain their causes in natural language.

3.1. Ontology-based representation of product line models

Figure 3 shows the proposed OWL ontology to represent the FMs concepts as an ontology. This representation allows us to exploit the semantic relationships among the concepts involved in FMs. For instance, we can ask for features that have the same parent, or features related by mandatory and exclude dependencies at the same time. We develop this ontology using the methodology proposed by Noy Noy & McGuinness (2001)and McGuinness [28] with a top-down approach, and we take the FM concepts from the meta-model presented in Figure 1.
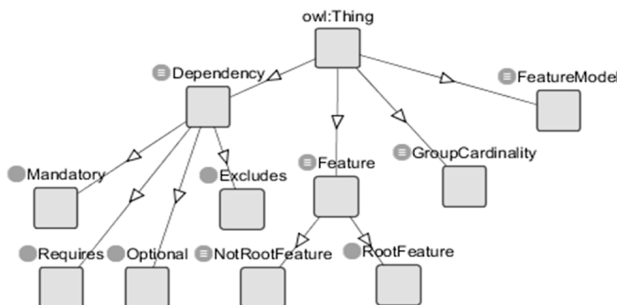

Figure 3. Proposed ontology to represent FMs

In our ontology, meta-model classes correspond to classes of the ontology. In addition, we separate the class *Feature* in classes *RootFeature* and *NotRootFeature* to represent that a FM only has one root feature. We represent the attributes of the *groupCardinality* meta-model class with ontology datatype properties, and we relate ontology classes with ontology object properties. For example, every dependency within the FM comprises an origin and a destination feature. Then, we create the object properties *hasDependencySource* and *hasDependencyDestination* to relate the *Dependency* ontology class with the *Feature* ontology class. Thus, we can relate each dependency with its feature source and its feature destination. Furthermore, in the optional and mandatory dependencies, the property *hasDependencySource* identifies the parent feature, and the property *hasDependencyDestination* identifies the child feature. In the example shown in Figure 2, the *Mobile Phone* feature is the origin feature of the optional dependency *R1*, and the *Utility Functions* feature is the destination feature. Moreover, *Mobile Phone* is the parent feature of the *Utility Functions* feature.

3.2. Rules for identifying dead features

According to the literature [4, 7, 13], misuse of dependency in FMs causes dead features. Our proposal considers that a feature can become a dead feature if a full-mandatory feature excludes an optional feature (see Rule 1), or if the parent feature is a dead feature (see Rule 2). Other cases that cause dead features are outside the scope of this initial proposal.

***Rule 1: Full-mandatory feature excludes an optional feature*:** An optional feature becomes dead when a full-mandatory feature excludes it.

In the example, the *Camera* feature is optional due to its dependency (R11) with the *Media* feature. Furthermore, product cannot have the *Games* and *Camera* features simultaneously due to the exclude dependency (R17) between both features. Because *Games* is a full-mandatory feature [5] (i.e., it is present in all products), the *Camera* feature is a dead feature.

***Rule 2: The parent feature is a dead feature*:** If a child feature is included during product configuration, the parent feature should be included too [3]. If the parent

feature is already a dead feature, its children features cannot be included in any product. In the example, features *VGA* and *MegaPixels* are children of dead feature *Camera*. Thus, these children features are dead features too.

We use SQWRL to implement the rules proposed in this section. For the sake of space, we only present and explain in the Table 2 the source code of the Rule 1, in which full- mandatory features exclude an optional feature. Nevertheless, both rules have a similar structure.

It is important to highlight that in queries, we use `WILDCARD` word as an argument that depends on each rule (e.g. in rule 1 `WILDCARD` belongs to full-mandatory features, while in rule 2 it belongs to dead features). Each statement SQWRL requires a constant value in the argument `WILDCARD`. Therefore, we create dynamically a SQWRL for each possible value of `WILDCARD`. For instance, `WILDCARD` can take seven different values in our application example (see Table 2); hence, we create seven different SQWRL queries.

Table 2. SQWRL query for dead features that satisfy Rule 1.

```
       (1)Excludes(?y) ^
       (2)Optional(?w) ^
     (3) NotRootFeature(?x) ^
    (4) NotRootFeature(WILDCARD) ^
   (5)hasDependencySource(?y, WILDCARD)^
(6)hasDependencyDestination(?y,?x) ^ (7)hasDependen
cyDestination(?w,?x)-> (8)sqwrl:selectDistinct(?x)
```

Rule 1: Consequent result

`selectDistinct(?x):`Optional feature, which is excluded by a full-mandatory feature.
Example Value:*Camera ,MSM*

Rule 1: Antecedent construction		
SQWRL instruction	**Definition**	**Example Value**
`Excludes(?y)`	Excludes dependencies	R16,R17,R18
`Optional(?w)`	Optional dependencies	R3,R9,R15,R19,R20
`NotRoot Feature (?x)`	Features non-root of the FM	Utility Functions, Settings, Calls, Messaging, Games, Java support, OS, Media, MP3,MP4, Camera,Voice, Data,SMS,MSM,VGA, Megapixels

`NotRoot Feature (WILDCARD)`	In this rule, `WILDCARD` correspond to full-mandatory features	Utility Functions, Settings, Calls, Messaging,Games, Java support, OS
`has Dependency Destination (?w,?x)`	Data are restricted, so *x* corresponds to features destination of optional dependencies	***Value of x*** Media, MP3, Camera,Voice,Data,SMS, MSM,VGA, Megapixels
`has Dependency Source (?y, WILDCARD)`	Data are restricted, so *y* corresponds to excludes dependencies whose source feature is full-mandatory	**Values of y** R16,R17
`has Dependency Destination (?y,?x)`	Data are restricted so **x** now corresponds to features excluded by the dependency *y*.	**Values of x** *Camera and MSM* Both are dead features

3.3. Natural Language Explanations

We have a predefined explanation text for each proposed rule to identify dead features. Then, after identifying the dead features that satisfy rules 1 or 2, we explain the defect in natural language, as follows:

a) We determine if the dead feature satisfies rule 1 or rule 2.
b) We generate the predefined text that explains rule 1 or rule 2 in natural language.

Text to explain rule 1 is *"Optional feature **featureName** is dead because the full-mandatory feature **fullMandatoryFeatureName** excludes it through the dependency **exclusionDependencyName"**.*

Text to explain rule 2 is "*Feature **featureName** is dead because **parentFeatureName**, its ancestor feature, is a dead feature too*"

c) We execute a new SQWRL query to get dependencies and features names related to the predefined text, which explains the dead feature.
d) We replace information from the FM at hand as needed in the predefined text. Figure 4 shows and example of each explanation applied to our application example.

3.4. Implementation Details

We implemented our approach in two stages. In the first

stage, we created the proposed ontology with Protégé 3.4.8 to represent concepts of the FMs meta-model. In the second stage, we developed a tool to integrate our proposed OWL ontology with Java. This integration allows us to manage and query information of each analyzed FM.

The implementation process of our second stage was as follows:

a) We read the proposed ontology in Java.

b) We use Jena [29] to populate and manage the ontology with information of the analyzed FM.

c) We use JESS library, as reasoner engine, to execute from Java SQWRL queries to identify dead features (i.e., features that satisfy the rule 1 or rule 2).

d) We produce a natural language text, which explains the cause of each dead feature. The explanatory text depends on whether the property satisfies rule 1 or rule 2. We complete the explanations from information gather from SQWRL queries.

Figure 4 presents one snapshot of the developed tool with a feedback obtained when we analyzed dead features in our application example. This case comprises features with mandatory, optional, excludes and group cardinality type dependencies. It also comprises an exclusive dependency that does not generate dead features (R18) and additional dependencies that generate dead features.
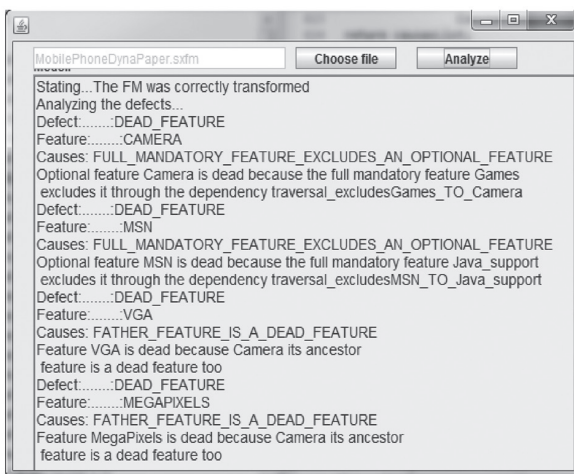
Figure 4. Snapshot corresponding to results generated from analyzing our FM running example

4. PRELIMINARY EVALUATION

4.1. Correctness

We assessed the correctness of our approach with 5 FMs with 25 features and 5 FM with 50 features. We generated these FMs with the BEnchmarking and TesTing on the analYsis (BeTTy) tool [30].

We manually tested our approach in three steps. First, we verified that it did not generate false positives. Second, we verified that the proposed solution identified 100% of dead features considered in our two rules. Finally, if the FMs had dead features, we validated that the cause corresponds to the case that produced the defect, and that the filled spaces in the explanation text corresponded to the correct situation for each one of the models.

In the first stage, we manually compared the dead features with the results obtained using FaMa [12]. We found that our proposal identified the 100% of the dead features that satisfied our rules, with 0% false positive. For the second and third stage, we made a manual inspection of correctness over 2 models (randomly selected) of 25 features and 2 models of 50 features.

We found that our proposal constructed correct explanations; i.e., they corresponded to the cause(s) that originated each defect. Results are available online in https://sites.google.com/site/raulmazo/.

4.2. Comparison of results

We compared results obtained in our proposal with the proposals of Trinidad *et al.* [4] and Rincón *et al.* [31] for the example application.

Table 3 presents the comparison of the results. The first column shows dead features identified by all approaches. The second column shows causes, in natural language, found by our proposal. Finally, the third column shows corrections proposed by Trinidad *et al.* [4] and Rincón *et al.* [31] (In this case, both approaches identified the same corrections).

Table 3. Comparison our proposal vs other approaches

Dead feature	Our proposal	Trinidad *et al.* [4] Rincón *et al.* [31]
	Causes in natural language	**Corrections**
Camera	*Optional feature* **Camera** *is dead because the full-mandatory feature* **Games** *excludes it through the dependency traversal_Games_TO_Camera*	R1
		R6
		R17
MSN	*Optional feature* **MSN** *is dead because the full-mandatory feature* **Java support** *excludes it through the dependency traversal_ MSN _TO_ Java support*	R2
		R7
		R16
Mega pixels	**Megapixels** *is dead feature because* **Camara** *its ancestor feature is dead feature too*	R1
		R6
		R17
		R19
VGA	*VGA is dead feature because* **Camara** *its ancestor feature is dead feature too*	R1
		R6
		R17
		R19

The results obtained shows that in the application example all approaches identified the same dead features. However, in other FMs, Trinidad *et al.* [4] and Rincón *et al.* [31] could identify other dead features that our approach will not identify. This is because we have not implemented all the cases to identify all dead features. Trinidad *et al.* [4] and Rincón *et al.* [31] identify all cases because they use a constraint satisfaction approach. However, our rule-based approach is extensible: we can create new rules for identifying and explaining in natural language other cases of dead features. Regarding explanations, Trinidad *et al.* [4] and Rincón *et al.* [31] identify the list of dependencies that must be deleted to remove dead features (Corrections). Our work instead focuses on explaining the cause of each dead feature in natural language. This information helps feature modelers to understand why dead features appear. Therefore, our approach is complementary to proposal of Trinidad *et al.* [4] and Rincón *et al.* [31] because the feature modeler could find dead features, their causes in natural language and possible corrections combining those proposals.

5. RELATED RESEARCH

We divide the research studies on identifying causes for dead features into two types: studies related to using ontologies in product line models, and those related to identifying causes of dead features. For the first type, Wang et al. [32] Wang et al., (2007)propose representing FMs and their constraints in OWL ontology language. In their proposal, the authors represent each feature as an ontology class, and each dependency as an ontology property. Their study identifies inconsistencies in particular FMs configurations and provide explanations for inconsistencies. However, their approach does not analyze the FM itself to identify the shortcomings, but each particular configuration.

In [15] Abo et al. propose two SWRL rules to validate model consistency. The first one detects features that excludes and requires the same feature, and the second one detects cycles in the FM, i.e, feature x requires feature y, and feature y requires feature x. Authors define inside ontology, as an antecedent, each situation that creates an inconsistency, and define as the consequence, the elements involved. Our work as the proposal of Abo et al. [15] uses ontology to represent FM in a formal way. However, additionally to use the ontology for formal representation, we use ontologies for two different purposes: (a) we exploit the ontological representation to perform dynamic SQWRL in FMs (i.e. Table 2); and (b) we explain defects in natural language. This is possible due to integration of our approach with Java. Abo et al. [15] implemented their proposal only in Protégé-OWL, therefore they do not have those advantages.

Lee *et al.* [33] Lee, Kim, Song, & Baik (2007) use ontologies to represent FMs and to analyze their variability and commonality. However, they use ontologies to analyze the semantic similarity of the FM, whereas that our approach uses ontologies to identify dead features and explain their causes.

Regarding the second category, several works were carried out to automatically identify dead features (and other defects) on [5, 9–11]. However, none of these works deals with identification of causes that explain in natural language why each dead feature occurs.

Trinidad *et al.* [4] present an automated method for identifying and explaining defects, such as dead features in FMs. For Trinidad *et al.* [4], an explanation is the minimal subset of dependencies that should be modified to remove the defect. They implemented their

approach transforming FMs into a diagnostic problem and then into a constraint satisfaction problem. This Implementation is available in FaMa [12], an Eclipse Plug-in for automatic analysis of FMs.

Rincón *et al.* [31] propose a method to identify corrections in FMs. In this approach, authors transform FMs into a constraint problem, then they identify all minimal corrections subsets (MCSes) [34] of dependencies that could be modified to correct each dead feature of the FM. This approach like FaMa[12], identify the list of dependencies that entail the fewest changes to fix the defect, but also identify others set of dependencies that imply more changes and fix the defect. This information provides more complete information about how to correct each dead feature.

Trinidad and Ruiz-Cortés [13] Trinidad & Ruiz-Cortes, (2009)use abductive reasoning to identify dead features and their causes. Unfortunately, the authors did not describe a method or algorithm to support their proposal.

Constraint satisfaction techniques are not enough to explain causes of a dead feature in natural language because, for instance, the structure needed to provide these explanations, is lost when authors transform the models into constraint programs. In fact, explanations generated by Trinidad *et al.*[4] and Rincón *et al.* [31] are not the causes that explain why a feature is dead, but corrections to apply in order to remove the defect.

6. CONCLUSIONS AND DISCUSSION

In this paper, we present an approach, which takes advantage of the inherent characteristics of ontologies, in order to identify dead features in FMs and explain certain causes of that defect in natural language. We use OWL to create an ontology to represent the FMs and their dependencies, and SQWRL as an ontology query language. We validate our proposal through the implementation and application of two SQWRL rules on a well-known case study and ten other FMs.

Our approach, in contrast to the black box-like approaches found in literature, can be easily extended with other rules to identify and explain other causes that create dead features.

Although ontologies were initially proposed for the semantic web, given their expressive power and formal semantics, they are useful in product lines to support identification of defects in feature models and to obtain information to produce explanations in a human understandable form.

We are currently extending this approach to identify other causes to explain dead features and other defects on FMs (e.g., false optional features or void FM). Other future directions include validating performance, accuracy, and scalability of the proposed approach for application to industrial cases.

ACKNOWLEDGMENTS

REFERENCES

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

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

[3] Kang, K.C., Cohen, S.G., Hess, J.A., Novak, W.E. and Peterson, S.P., Feasibility Study Feature-Oriented Domain Analysis (FODA). Technical Report 1990.

[4] Trinidad, P., Benavides, D., Duran, A., Ruiz-Cortés, A. and Toro, M., Automated Error Analysis for the Agilization of Feature Modeling. Journal of Systems and Software., 81, pp. 883–896, 2008.

[5] Von Der Massen, T. and Lichter, H., Deficiencies in Feature Models , En: Workshop on Software Variability Management for Product Derivation - Towards Tool Support. (Eds.Mannisto T, Bosch J). 2004.

[6] Trinidad, P., Benavides, D. and Ruiz-Cortés, A., Isolated Features Detection in Feature Models , En: Proceedings of Conference on Advanced Information Systems Engineering (CAiSE 2006). pp. 1–4, 2006.

[7] Benavides, D., Segura, S. and Ruiz-Cortés, A., Automated analysis of feature models 20 years later: A literature review. Information Systems., 35, pp. 615–636, 2010.

[8] Sun, J., Zhang, H. and Wang, H., Formal Semantics and Verification for Feature Modeling , En: Proceedings of the 10th IEEE International Conference on Engineering of Complex Computer Systems. IEEE Computer Society, Washington, DC, USA, pp. 303–312, 2005.

[9] Salinesi, C. and Mazo, R., Defects in Product Line Models and how to identify them , En: Software Product Line - Advanced Topic.(Eds.Elfaki A). pp. 97–122, 2012.

[10] Thüm, T., Kastner, C., Benduhn, F. and Meinicke, J., SAAK: FeatureIDE: An extensible framework for feature-oriented software development. Science of Computer Programming., 2012.

[11] Mazo, R., Salinesi, C. and Diaz, D., VariaMos : a Tool for Product Line Driven Systems , En: Proceedings of the 24th International Conference on Advanced Information Systems Engineering (CAiSE Forum'12). Springer Press, Gdansk-Poland, pp. 25–29, 2012.

[12] Trinidad, P., Benavides, D., Ruiz-Cortés, A., Segura, S. and Jimenez, A., FAMA Framework , En: Proceedings of the 12th International Software Product Line Conference. IEEE Computer Society, Washington, DC, USA, 359, 2008.

[13] Trinidad, P. and Ruiz-Cortés, A., Abductive Reasoning and Automated Analysis of Feature models: How are they connected , En: Proceedings of the Third International Workshop on Variability Modelling of Software-Intensive Systems. pp. 145–153, 2009.

[14] Batory, D., Benavides, D. and Ruiz-Cortés, A., Automated analysis of feature models: challenges ahead. Communications of the ACM., 49, pp. 45–47, 2006.

[15] Abo, L., Kleinermann, F. and De Troyer, O., Applying semantic web technology to feature modeling , En: Proceedings of the 2009 ACM symposium on Applied Computing (SAC '09). ACM, New York, pp. 1252–1256, 2009.

[16] Czarnecki, K., Peter Kim, C.H. and Kalleberg, K.T., Feature Models are Views on Ontologies , En: Proceedings of the 10th International on Software Product Line Conference. IEEE Computer Society, Washington, DC, USA, pp. 41–51, 2006.

[17] Mcguinness, D.L. and Van Harmelen, F., OTHERS: OWL web ontology language overview. W3C recommendation., 10, 10, 2004.

[18] O'connor, M., and Das, A., SQWRL: a Query Language for OWL , En: Proceedings of the 6th International Workshop OWL: Experiences and Directions. Chantilly, 2009.

[19] Tsetsos, V., Papataxiarhis, V. and Hadjiefthymiades, S., Personalization based on Semantic Web Technologies. Semantic Web Engineering in the Knowledge Society, Information Science Reference., pp. 52–75, 2008.

[20] Breitman, K.K., Casanova, M.A. and Truszkowski, W., Semantic web: concepts, technologies and applications Springer-Verlag, pp. 105–127, 2007.

[21] Djebbi, O. and Salinesi, C., Criteria for Comparing Requirements Variability Modeling Notations for Product Lines , En: Proceedings of the Fourth International Workshop on Comparative Evaluation in Requirements Engineering. IEEE, pp. 20–35, 2006.

[22] Czarnecki, K., Helsen, S. and Eisenecker, U., Formalizing Cardinality-based Feature Models and their Specialization. Software Process: Improvement and Practice., 10, pp. 7–29, 2005.

[23] Segura, S., Automated Analysis of Feature Models Using Atomic Sets , En: Proceedings of the First Workshop on Analyses of Software Product Lines (ASPL08). 2008.

[24] Borst, W., Construction of Engineering Ontologies for Knowledge Sharing and Reuse: Ph.D. Dissertation [Ph Thesis].Enschede, Netherlands: University of Twente,1998.

[25] Gruber, T., Toward Principles for the Design of Ontologies Used for Knowledge Sharing , En: International Workshop on Formal Ontology.(Eds.Guarino N, Poli R). Padova,Italy, 1993.

[26] Horridge, M., Knublauch, H., Rector, A., Stevens, R. and Wroe, C., A Practical Guide To Building OWL Ontologies Using The Protege-OWL Plugin and CO-ODE Tools Edition 1.02004.

[27] Friedman-Hill, E., Java expert system shell, Available:http://herzberg.ca.sandia.gov/jess.

[28] Noy, N. and Mcguinness, D., Ontology Development 101 : A Guide to Creating Your First Ontology, pp. 1–25, 2001.

[29] APACHE SOFTWARE FOUNDATION: Apache Jena, Available:http://jena.apache.org/documentation/ontology/.

[30] Segura, S., Galindo, J.A., Benavides, D., Parejo, J.A.

and Ruiz-Cortés, A., BeTTy: benchmarking and testing on the automated analysis of feature models , En: Proceedings of the Sixth International Workshop on Variability Modeling of Software-Intensive Systems. ACM, New York, NY, USA, pp. 63–71, 2012.

[31] Rincón, L.F., Giraldo, G.L., Mazo, R., Salinesi, C. and Diaz, D., Subconjuntos Minimos de Correccion para explicar caracteristicas muertas en Modelos de Lineas de Productos. El caso de los Modelos de Caracteristicas , En: Proceedings of the 8th Colombian Computer Conference (CCC). Armenia-Colombia, 2013.

[32] Wang, H., Li, Y., Sun, J., Zhang, H. and Pan, J., Verifying feature models using OWL. Web Semantics: Science, Services and Agents on the World Wide Web., 5, pp. 117–129, 2007.

[33] Lee, S., Kim, J., Song, C. and Baik, D., An Approach to Analyzing Commonality and Variability of Features using Ontology in a Software Product Line Engineering , En: Proceedings of the Fifth International Conference on Software Engineering Research, Management and Applications. IEEE, pp. 727–734, 2007.

[34] Liffiton, M. and Sakallah, K., Algorithms for Computing Minimal Unsatisfiable Subsets of Constraints. Journal of Automated Reasoning., 40, pp. 1–33, 2008.