Semantic Enrichment and Verification of Feature Models in DSPL

Thalisson Oliveira, Rossana M. C. Andrade[*] and Windson Viana
Group of Computer Networks, Software Engineering and Systems (GREat), Federal University of Ceará, Fortaleza, Ceará, Brazil

Keywords: Dynamic Software Product Line, Context-awareness, Feature Model, Automatic Verification Tool, Ubiquitous Software, Semantic Enrichment.

Abstract: Dynamic Software Product Lines (DSPLs) support the development of context-aware systems, which use context information to perform adapted services aiming to satisfy user's needs. Feature models (FM) represent system similarities and variability in DSPL. However, some FM representations are limited in expressiveness. For example, relevant domain aspects (e.g., context-aware feature that implements a particular use case) are not described in FM. This research proposes an approach based on an OWL-DL ontology to add semantics to FM. It also provides automatic verification of the correctness and consistency of these models. We implemented this approach in a feature model design tool called FixOnto. Our first evaluation results showed that the use of ontologies brings benefits such as improvements on SPL information retrieval, and inference and traceability of the features, use contexts, and SPL artifacts.

1 INTRODUCTION

Currently, powerful mobile devices with several sensors enable the development of context-aware applications, i.e., applications that can perceive the environment and adapt themselves to satisfy the user's needs (Marinho et al., 2013). However, due to self-adaptation complexity, the development of context-aware systems becomes a challenging activity. Software engineers have to predict context variations, including exceptional situations and have to code consistent adaptation methods (Marinho et al., 2012).

Software Engineers can use SPL to systematize and maximize reuse in context-aware systems development (Marinho et al., 2013). Dynamic Software Product Lines (DSPLs) are SPLs that include mechanisms to change variants in self-adaptive systems at runtime (Hallsteinsen et al., 2008), supporting context-aware systems development. In (Marinho et al., 2012), authors propose the concept of Context-Aware Software Product Lines (CASPL), a subset of DSPL. They introduce Context-Aware Feature Models (CAFM). A CAFM is composed of a system model (SM) and a context model (CM). CASPL are our research target.

In CASPL, feature and context modeling are the main activities involved in their design (Dermeval et al., 2015). CAFM represents feature variability and also predefined context situations and adaptations. Context-aware systems modeling is an error-prone task, as a consequence, a CASPL can derive invalid products. Thereby, verification of CAFM is a necessary step. However, manual verification in complex systems, such as context-aware systems, and in large-scale is infeasible (Benavides et al., 2013). Thus, there is a necessity for automatic verification of feature models to prevent inconsistencies.

To motivate our approach, let us take the domain of Mobile Visit Guides (MVG). We considered it as a family of self-adaptive products, in which applications can display content (e.g., text, image, and video) according to the user's location, and his device configurations and status. In our example, the domain expert should associate Video Display feature to user's location and his device battery level. To perform this kind of adaptation, the domain expert must select the Message Exchange feature since he has modeled it as a mandatory feature. However, the domain expert can model a context situation that not choose the Message Exchange feature, then, the SPL will derivate a product with an inconsistent adaptation according to the FM.

Regarding this scenario, this work proposes an approach and a tool for adding semantic and automatic verification of correctness and consistency of CAFM.

[*]CNPq Productivity Scholarship in Technological Development and Innovative Extension (DT-2)

OWL-DL ontologies [2] and inference mechanisms are the core assets of our approach. The use of ontologies adds new capabilities to DSPL such as improvements on information retrieval and traceability. In our approach, a DSPL modeler can perform domain modeling, associate core assets to the FM, and also performs FM verification.

We evaluated our tool with the following research questions (RQ) in mind. RQ1: How to semantically enrich a DSPL (associating features and contexts to core assets) and ensure that the mapping remains correct after this association? RQ2: How to make information retrieval and traceability in DSPL easier?

2 BACKGROUND

As the main goal of our research is to provide a solution for semantic enrichment and CAFM verification, firstly, details of CASPL and CAFM are shown in this section. Secondly, semantic enrichment approaches are presented. Finally, CAFM verification is discussed.

2.1 Context-aware SPL

Context-aware SPL focus on deriving products that can provide services according to user's needs, observing the environment around them (Costa et al., 2015). Thus, CASPL is an SPL that promote reuse in the development of context-aware applications.

A key component of CASPL is the Context-aware Feature Model (CAFM), which combines traditional feature model with a context model in order to provide a better SPL description. CAFM is composed by four diagrams: (i) System Model (SM), representing SPL similarities and variability, (ii) Context Model (CM), representing the known contexts, (iii) composition rules (CR), related to SM and (iv) adaptation rules (AR) or context rules, related to CM (Marinho et al., 2012).

System Model is a tree-like diagram containing a unique root, which represents the domain. The reminder nodes correspond to domain features and edges describe the hierarchical relationships between these features (Marinho et al., 2012). One example of SM is found in Figure 1 (a). MobileGuide root node corresponds to the domain being represented. Message Exchange node is a mandatory feature. Exchange Type node is a variation point, which can be Asynchronous or Synchronous. Security and Privacy are optional features.

[2]http://www.w3.org/TR/owl-features/

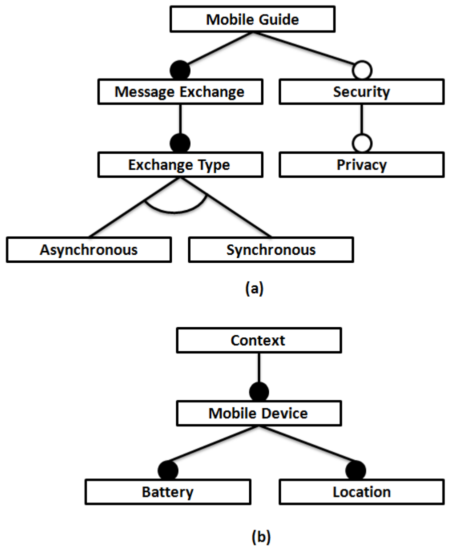


Figure 1: MobiLine (a) System Model fragment and (b) Context Model fragment.

Context Model has four levels. The first level is the modeled context and contains only one node. Nodes in the second level represent the Context Entities and nodes in both third and fourth levels show Context Information and Context Attributes, respectively (Marinho et al., 2012). In Figure 1 (b) there is an example of CM. The root node Context represents one context that needs to perform adaptations through context entity. Context entity (MobileDevice) is the entity observed by the context in question, and it provides context information. It represents which entity information will be read (e.g., battery level and Location). In the fourth level, there are the context attributes, which are possible values to context information (e.g., battery level can be 0 to 100%) and are not represented in the figure.

Composition rule is formally specified as an implication of an antecedent expression to a consequent expression, being each expression one propositional formula over a set of features and attributes contained in the System Model (Marinho et al., 2012). In Table (1), an example of rule expression is presented. The rule CR1 shows that the presence of Synchronous ExchangeType implies in the use of Security feature.

Adaptation rule also is formally specified in (Marinho et al., 2012), as an implication of a context expression (CE) to a system expression (SE). An event can be a state or combination of states of a system. One action can also be a new state or a combination of states in the system. For instance, if the battery level reaches a value lower than 10%, the application disables the video displaying. In (1), one example of adaptation rule expression is presented. The rule AR1 means than if the location of the device is an insecure area implies in the use of Privacy feature.

Table 1: Example of Rules proposed in (Marinho et al., 2012).

Rule	Expression
CR1	$ExchangeType.Synchronous \implies Security$
AR1	$(MobileDevice.Location = UnsecureArea) \implies Privacy$

One example of CASPL is MobiLine (Marinho et al., 2013), which is a CASPL for the mobile and context-aware domain. This domain holds adaptability as a principle and suggests that during execution time, the applications should be able to adapt themselves according to the changes in user's context (Marinho et al., 2013). Figure 1 contains a fragment of Mobiline (Marinho et al., 2013), which is an SPL for the domain of mobile and context-aware applications. For instance, the Mobilide Guide represents the domain, and Message Exchange feature is necessary to display information in the mobile guide.

The CASPL concepts explained in this section guided the solution proposed in this work. We used MobiLine as case study and examples showed in Figure 1, illustrated system model and context model in CAFM and the Extended FM notation (Benavides et al., 2013) was used. Figure 1, showed previously, illustrated system model and context model in CAFM and the Extended FM notation (Benavides et al., 2013) was used.

2.2 Adding Semantics to SPL

Software Product Lines are well known for providing a high level of reuse. SPL core assets contain the set of artifacts used to build a product. Feature models are commonly used to represent SPL and can be used to group artifacts from core assets and help on traceability. In (Filho et al., 2012), the authors introduce the concept of Semantic Software Product Line (SSPL) as an SPL where reusable core assets and business artifacts are related to the domain model and can be expressed using one or more ontologies. Thus, ontologies increase knowledge expressiveness in SPL and allow infer new information and implicit relations by using inference algorithms provided by reasoners.

Feature model can be a start point for adding semantic to an SPL. In our work, add semantics means to associate core assets to FM, making this FM more expressive. For instance, relationships between a feature and other SPL artifacts (e.g., source code and requirements documents) can be specified. The authors of (Filho et al., 2012) propose a process that involves two main steps: (i) transformation of a feature model in an ontology (OWL DL), performed automatically by a tool called Fea2Onto, and (ii) the addition of new relations with external artifacts in a process guided by a top ontology called SPLiSEM. When the process is finished, a more expressive model of SPL is produced.

2.3 Feature Model Verification

Feature Models (FM) are widely used to represent SPL, and it is relevant to assure the model correctness and consistency. Regarding model verification of CASPL, (Marinho et al., 2012) propose a formal process to check correctness and consistency of a Context-aware Feature Model (CAFM). That process describes a set of well-formedness rules for Feature Model and Context Model. They also provide verification if the model is in accordance with composition and adaptation rules. In Table 2, the description of a subset of Well Formed Composition Rules (WFCR), Well Formed Adaptation Rule (WFAR) and Inter-rules Consistency (IRC) proposed in (Marinho et al., 2012) are listed.

Table 2: Subset of Rules proposed in (Marinho et al., 2012).

Rule	Description
WFCR1	Features referenced in a Composition Rule should be either an optional feature or an attribute feature.
WFAR1	Features referenced in the System Expression should be either an optional feature or an attribute feature and should be owned by the SM
IRC1	CRs defined for a CAFM should be consistent with each other

Based on (Marinho et al., 2012), a tool was developed in (Costa et al., 2015) and provides CAFM modeling, verification of feature model well-formedness and consistency and also a simulation of product derivation. The tool in (Costa et al., 2015), Marinho *et. al* approach (Marinho et al., 2012), and semantic enrichment approach (Filho et al., 2012) were the start point of our proposal.

3 AN APPROACH FOR SEMANTIC ENRICHMENT AND VERIFICATION OF CAFM

This work proposes an approach that permits context-aware feature modeling, semantic enrichment of these models, and automatic feature model verification. Like a traditional SPL, CASPL also takes benefits from the integration of semantic enrichment and model verification. We aim at assist CASPL modelers and all stakeholders involved in the context-aware applications development.

3.1 Design Principles

Our approach is composed of four main steps: (a) feature modeling, which includes semantic enrichment, (b) automatic model transformation, (c) automatic verification of feature model, and (d) semantic search.

In the first step (Feature Modeling and Semantic Enrichment), a domain specialist performs feature modeling, context modeling, and associates these models to domain artifacts. In the feature modeling process, the constraints between features and context situations are modeled. A CAFM is the output of this first step.

In the second phase (Automatic Mapping), the CAFM is transformed in an OWL DL ontology automatically. This ontology represents features, contexts, rules, and artifacts present in the CAFM. In this process, the transformation requires a top ontology containing CAFM concepts (e.g., feature, context situations, rules). The section 3.2 details this top ontology.

A reasoner analyzes the OWL version of the CAFM in the “Model Verification” phase. The reasoner evaluates the model correctness and consistency. This verification uses the rules proposed in (Marinho et al., 2012). For instance, if the ontology is valid (all derived product obey the rules), then it is ready to last step (semantic search).

The fourth step (Semantic Search) allows to domain specialist performing queries over the improved models. It can retrieve feature’ information and traces, relationships between context scenarios and model artifacts. An ontology reasoner is a support for this kind of query. One example is the DL Query language provided by Protégé [3].

We implemented a tool called FixOnto, which permits both feature and context modeling. FixOnto also allows semantic enrichment making association among features and both use case and test cases. Moreover, it is possible to perform model verification and information retrieval.

3.2 Top Ontology for CAFM

Figure 2 shows a fragment of the top ontology representing the CASPL. The System Model (SM) accounts for the variability and commonalities between features of the domain modeled. Thus, it is important that concepts like (*mandatory feature*) and (*optional feature*) are presented in the ontology. The context is a specialization of the system model, which represents the context situations in the domain, containing entities and contextual information. Therefore, the ontology also contains these concepts.

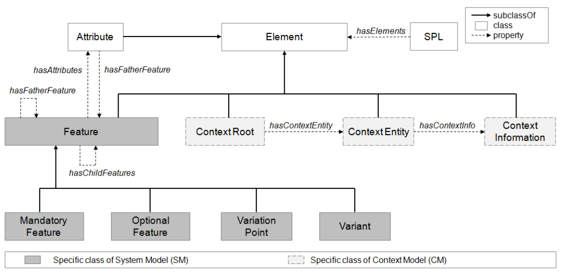


Figure 2: Top ontology representing System and Context Models in CASPL.

CASPL is composed by elements, which are related to system diagrams and context diagrams (Figure 2). The system model consists of a feature hierarchy. In this top ontology, the *Feature* subclasses can have self-relationships. For instance, an individual of *Mandatory Feature* can have a subfeature and individual of *Variation Point*. The opposite is also possible. The domain specialist relates features to other features by using the properties *hasFatherFeature* and *hasChildFeatures*. The first property indicates that feature C is a subfeature of B, and we can say that *hasFatherFeature(C,B)*. The second property indicates that B has the child feature C. The axiom *hasChildFeatures(B,C)* represents it.

The context model has a *ContextRoot*, which represents the context variation in which applications need to perform adaptations. *ContextEntity* is the object or entity that is observed by the context in question, for instance, a mobile device. Context information indicates which information the system will read since not all elements provided by the context entity are relevant to context-aware applications. The battery level and the Wi-Fi signal strength are examples of context information that can be captured by a mobile device.

Figure 3 shows the composition rules and adaptation rules that compose a CASLP. They are represented by *Composition Rule* and *Context Rule* classes respectively.

3.3 Verification Rules of CAFM

The third process of our approach is the CAFM verification. An OWL ontology is the input of this process.

[3]http://protege.stanford.edu

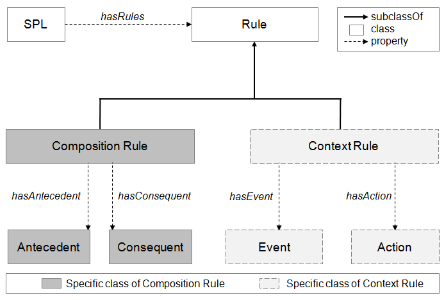


Figure 3: Fragment of Top ontology representing the Composition Rules and Adaptation Rules in a CASLP.

In this step, we verified if the ontology obeys a subset of well-formedness and consistency rules, which were proposed in (Marinho et al., 2012). To do this, we represent these rules in SWRL.

An SWRL rule is composed of a body expression that implicates a head expression. If the body expression is reached the head sentence is executed. We mapped the composition and adaptation rules writting the SWRL body as an inconsistent state of model (the state of ontology individuals that means a broken rule). We created OWL classes and individual properties that represents each type of inconsistency to write the head expressions. Thus, when an ontology individual reaches an inconsistent state, it is highlighted associated to corresponding OWL class error.

In the expression $CompositionLiteral\,(?x) \wedge MandatoryFeature\,(?y) \wedge hasFeaturedElement\,(?x,?y) \rightarrow WFCR1\,(?x)$, the rule WFCR1 indicates which an action related to a mandatory feature is an exceptional situation. In case this inconsistency occurs the individual will be highlighted as a WFCR1 type. The same meaning is applicable to the expression $ActionLiteral\,(?x) \wedge MandatoryFeature\,(?y) \wedge hasFeaturedElement\,(?x,?y) \rightarrow WFAR1\,(?x)$. If a mandatory feature y is present in a System Expression on adaptation rules, it will be highlighted as a WFAR1 individual. Therefore, it is possible to identify inconsistencies in the ontology after a reasoner analysis.

4 FixOnto TOOL

We implemented a tool to help domain expert with the enrichment and verification process. FixOnto tool allows CAFM modeling and semantic enrichment, mapping CAFM into OWL files, model verification, and semantic search. Figure 4 shows a screenshot of tool.

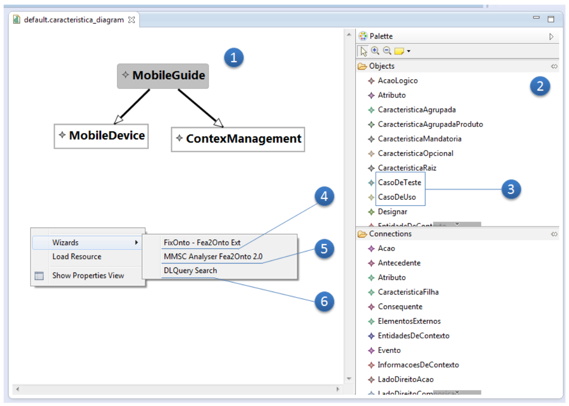


Figure 4: Fragment of FixOnto

4.1 CAFM Modeling and Semantic Enrichment

FixOnto is an extension of the tool proposed in (Costa et al., 2015). The modeling editor on the original software was the start point towards FixOnto implementation. We inherited the tool area to graphically model CAFM represented by (1) and the pallet represented by (2) in Figure 4. We added concepts to assist domain experts in the SPL semantic enrichment process. The options "TestCase" and "UseCase" represented by (3) in Figure 4 can be associated with features and context situations, making the model a richer CAFM.

The FixOnto is an Eclipse-based tool, generated from UML-like models using Eclipse Modeling Framework (EMF) and Eclipse Epsilon [4]. By the use of Epsilon, mapped CAFM can work with Java, the programming language used to code the others steps in FixOnto.

4.2 Automatic Mapping

By using FixOnto, domain specialist needs only to perform a right-click on feature modeling area, select option "wizard" and select "FixOnto - Fea2Onto Ext" to translate the model into OWL, such as represented by (4) in Figure 4. This option will generate the file fixonto.owl, representing a rich CAFM that is ready to be analyzed.

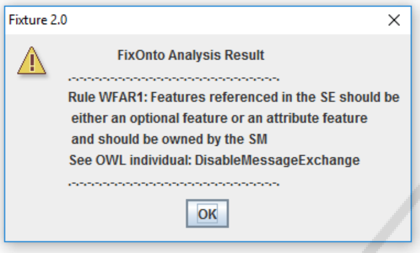
FixOnto CAFM transformation is inspired by the Fea2Onto tool proposed in (Filho et al., 2012). Fea2Onto steps are: (i) Java objects generation from the feature model, and (ii) ontology generation from objects, using OWL Java library and guided by a top ontology. In the FixOnto, we coded a Fea2Onto extension, which is a new implementation that including

[4]http://www.eclipse.org/epsilon/

context-awareness concepts.

4.3 Model Verification and Semantic Search

To perform model verification, the user should select option “CAFM Analyser”, represented by (5) in Figure 4. The FixOnto searches for a file named fixonto.owl and performs an update including SWRL rules contained in a file named rules.fix. After update fixonto.owl, our tool calls the Pellet reasoner that interprets and checks the SWRL rules. Among several reasoners, we chose Pellet because of its rule support (SWRL) and its free license (Dentler et al., 2011).


Fixture 2.0

FixOnto Analysis Result

Rule WFAR1: Features referenced in the SE should be either an optional feature or an attribute feature and should be owned by the SM

See OWL individual: DisableMessageExchange

Figure 5: FixOnto broken rule message.

At last, after Pellet reasoning, FixOnto displays a box containing found errors. In Figure 5, we can see violated rules and OWL element to be verified.

Once ontology is enriched and valid, it is ready to semantic search phase. Into FixOnto, domain experts are able to make semantic searches using Protégé Query Syntax, and the OWL individuals are presented. Figure 6 shows an example of information recovering (details in section 5.1).

5 EVALUATION

We modeled the CASPL Mobiline (Marinho et al., 2013) into FixOnto to show benefits of our approach and examine its drawbacks. To control the results, we selected only a part of Mobiline, represented in Figure 1. The modeling example followed the steps presented in the sequence of this paper.

5.1 Modeling a CASPL into the Tool

Firstly, we modeled the diagrams with FixOnto. Figure 1 placed in section 2.1 shows the System Model and the Context Model used in this tool execution. The expressions (1) and (2) in the section 2.1 are the modeled rules. We included illustrative Test Cases (TC1, TC2, and TC3) for increasing the CASPL expressivity.

After the initial modeling phase, FixOnto transformed the CAFM in OWL-DL ontology. The generated ontology was stored in a file called *fixonto.owl*. Then, we verified the model correctness and consistency by using well-formed rules. The modeled CAFM did not present errors. We, then, inserted inconsistencies into the model to check if error detection worked. Section 5.2 discusses in details the process of fault insertions.

Finally, we performed a semantic search over the ontology (fixonto.owl), using DL Query. We selected the option DLQuery Search in FixOnto. We queried about test cases existent in the model. It may be important to developers verify the scope of a software evolution. The queries and results can be viewed in Figure 6.

5.2 Fault Insertion

We inserted errors corresponding to rules viewed in Table 2 in section 2.3 and also extras rules. We changed some axioms of fixonto.owl file. In the first example, the original axiom was *hasFeaturedElement*(DisableMessage, ExchangeType.Synchronous). The ExchangeType.Synchronous feature is an alternative feature. In this case, the rule WFAR1 in Table 2 is not broken.

For instance, in order to insert an error, we changed the consequent of the context rule. We removed the consequent ExchangeType.Synchronous and, after, we included a mandatory feature (i.e., MessageExchange). The generated axiom after our change was *hasFeaturedElement*(DisableMessage, ExchangeType.Synchronous). Figure 5 shows the tool result. It shows the broken rule.

Other implemented rule is represented by expression $hasFatherFeature(?x,?y) \wedge hasFatherFeature(?y,?x) \rightarrow hasCycle(?x,?y) \wedge WFSMR4(?x)$. If a feature x is father of y and y is father of x, it is said a cycle, and x is highlighted as an individual of WFSMR4 error and the individuals x and y are associated by the property *hasCycle*. In this case, the original axiom was hasFatherFeature(Privacy, Security). We changed the axiom to $hasFatherFeature(Security, Security)$, and the error was thrown.

We implemented 12 rules and inserted 2 faults per rule, resulting in 24 faults. The percentage of fault detection for the implemented rules was 100%.

5.3 Discussion

This research proposes a solution for maximizing reuse. We aim at helping domain experts to model and

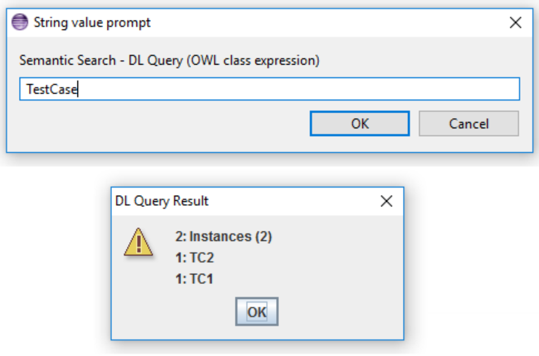


Figure 6: FixOnto Query Search and Result.

retrieve information on CASPL development. The proposed solution also includes model verification, helping to prevent modeling errors and reduce costs. The tool in (Costa et al., 2015), Marinho *et. al* approach (Marinho et al., 2012) and semantic enrichment approach (Filho et al., 2012) were the start point of this proposal.

Regarding the first research question (RQ1: How to semantically enrich a DSPL and ensure that the mapping remains correct after this association?), the FixOnto provides an automatic model verification the can be performed after semantic enrichment. If the verification does not throw errors, we ensure that mapping remains correct.

Concerning the second research question (RQ2: How to make information retrieval and traceability in DSPL easier?), we were able to trace the modeled Test Cases, by using "Query Search" FixOnto functionality. A drawback is that domain specialist needs to learn the DL Query syntax.

This research generated results that highlight the benefits of performing a semantic search over an expressive and valid CAFM ontology. The information retrieval and traceability allow to the domain modeler and other stakeholders take important decisions along the development of context-aware applications. However, some threats to validity were detected. We categorized the threats to validity using the following classification proposed in (Wohlin et al., 2012): Conclusion, Internal, Construct and External. We identified threats to Conclusion and External validity as follows.

Only one modeler used the FixOnto tool to verify its functionality implying in its threat to Conclusion validity, which makes necessary to do an experiment with a greater number of context-aware specialists. Moreover, we demonstrate the tool by using a single academic CASPL, restricting the results to its scope. This choice represents its threat to External validity. The modeling should be performed with more CASPL to generalize the results.

6 RELATED WORK

The proposal in (Filho et al., 2012) provides a solution to add semantic to SPLs. The authors use an ontology to represent a feature model and use it as start points to semantic enrichment. Narwane *et al.* (Narwane et al., 2016) focus on formal modeling and analysis of traceability in an SPL. Their work involves the relation between features and core assets.

Regarding feature model verification, Rincon *et al.* (Rincón et al., 2014) propose a rule based approach using ontologies to analyse *dead features* and *false optional* in feature diagrams. Their proposal identifies causes of inconsistencies and explaining them in natural language. In (Zaid et al., 2009), authors provide an ontology-based framework for feature modeling with a rule-based model to check consistency and detect conflicts.

All these related work do not address contextual aspects, which is the key point of our paper. In fact, our approach is an extension of (Filho et al., 2012) research. As previously discussed, the authors in (Marinho et al., 2012) propose an approach to context-aware model verification and the authors in (Costa et al., 2015) propose a tool that implements the approach. Thus, they address CAFM modeling, verification of feature model well-formedness and consistency. However, unlike our work, their proposals do not address semantic enrichment, which relates the model to the core assets.

7 CONCLUSIONS AND FUTURE WORK

Information retrieval and traceability help stakeholders to take a decision during the development of context-aware systems. Our research motivation has been both the importance of the domain information retrieval in CAFM and the importance of consistency verification of a DSPL (i.e. CASPL) during the project initial steps. Early detection of modeling inconsistencies allows time and money economy during software development cycle. In the literature, it is possible to find solutions that verify traditional SPL and DSPL, however, using these existing solutions, the information in a feature model can still be insufficient. To fulfill this gap, we proposed an ontology-based approach and a tool for adding semantic in CASPL and for verifying its correctness and consistency automatically.

For the proposition of our approach and respective tool, we combine some existing solutions. For semantic enrichment, the approach proposed by (Filho et al.,

2012) was one start point for our work. For automatic verification in CASPL, we used the framework proposed in (Zaid et al., 2009), which suggests the implementation of rules in SWRL. The rules into FixOnto were then implemented in SWRL following the rules proposed by (Marinho et al., 2012).

We demonstrated the semantic enrichment and automatic verification of CAFM with a tool called FixOnto, which is an extension of Fixture (Costa et al., 2015). We also demonstrated the traceability, which is very important to prevent errors and provide time efficiency, using the model with DL Query.

There are evidences that our approach can be used for traditional FM, but we still need to evaluate that in future work as well as we aim to automatize the whole approach. A usability evaluation should also be performed over FixOnto to verify the benefits to the final user, specifically the domain specialist. It is also important to model a second CASPL into the tool to analyze different scenarios. We also aim to increase the tool usability, displaying only human readable messages and excluding OWL individuals information.

REFERENCES

Benavides, D., Felfernig, A., Galindo, J. A., and Reinfrank, F. (2013). *Automated Analysis in Feature Modelling and Product Configuration*, pages 160–175. Springer Berlin Heidelberg, Berlin, Heidelberg.

Costa, P. A. d. S., Marinho, F. G., Andrade, R. M. d. C., and Oliveira, T. (2015). Fixture - A tool for automatic inconsistencies detection in context-aware SPL. In *ICEIS 2015 - Proceedings of the 17th International Conference on Enterprise Information Systems, Volume 2, April, 2015*, pages 114–125.

Dentler, K., Cornet, R., ten Teije, A., and de Keizer, N. (2011). Comparison of reasoners for large ontologies in the owl 2 el profile. *Semant. web*, 2(2):71–87.

Dermeval, D., Tenrio, T., Bittencourt, I. I., Silva, A., Isotani, S., and Ribeiro, M. (2015). Ontology-based feature modeling: An empirical study in changing scenarios. *Expert Systems with Applications*, 42(11):4950 – 4964.

Filho, J. a. B. F., Barais, O., Baudry, B., Viana, W., and Andrade, R. M. C. (2012). An Approach for Semantic Enrichment of Software Product Lines. *Proceedings of SPLC 2012*, II.

Hallsteinsen, S., Hinchey, M., Park, S., and Schmid, K. (2008). Dynamic software product lines. *Computer*, 41(4):93–95.

Marinho, F. G., Andrade, R. M., Werner, C., Viana, W., Maia, M. E., Rocha, L. S., Teixeira, E., Filho, J. B. F., Dantas, V. L., Lima, F., and Aguiar, S. (2013). Mobiline: A nested software product line for the domain of mobile and context-aware applications. *Science of Computer Programming*, 78(12):2381 – 2398. Special Section on International Software Product Line Conference 2010 and Fundamentals of Software Engineering (selected papers of {FSEN} 2011).

Marinho, F. G., Maia, P. H. M., Andrade, R. M. C., Vidal, V. M. P., Costa, P. A. S., and Werner, C. (2012). Safe adaptation in context-aware feature models. In *Proceedings of the 4th International Workshop on Feature-Oriented Software Development*, FOSD ’12, pages 54–61, New York, NY, USA. ACM.

Narwane, G. K., Galindo, J. A., Krishna, S. N., Benavides, D., Millo, J., and Ramesh, S. (2016). Traceability analyses between features and assets in software product lines. *Entropy*, 18(8):269.

Rincón, L., Giraldo, G., Mazo, R., and Salinesi, C. (2014). An ontological rule-based approach for analyzing dead and false optional features in feature models. *Electronic Notes in Theoretical Computer Science*, 302(0):111 – 132. Proceedings of the {XXXIX} Latin American Computing Conference (CLEI 2013).

Wohlin, C., Runeson, P., Hst, M., Ohlsson, M. C., Regnell, B., and Wessln, A. (2012). *Experimentation in Software Engineering*. Springer Publishing Company, Incorporated.

Zaid, L. A., Kleinermann, F., and De Troyer, O. (2009). Applying semantic web technology to feature modeling. In *Proceedings of the 2009 ACM symposium on Applied Computing*, pages 1252–1256, New York, NY, USA. ACM.