Ontology-based feature modeling: an empirical study in changing scenarios

http://www.producao.usp.br/handle/BDPI/50184

Downloaded from Biblioteca Digital da Produção Intelectual - BDPI, Universidade de São Paulo
Ontology-based feature modeling: An empirical study in changing scenarios

Diego Dermeval\textsuperscript{a,b,*}, Thyago Tenório\textsuperscript{c}, Ig Ibert Bittencourt \textsuperscript{c}, Alan Silva \textsuperscript{c}, Seiji Isotani \textsuperscript{d}, Márcio Ribeiro \textsuperscript{c}

\textsuperscript{a}Computing and Systems Department, Federal University of Campina Grande (UFCG), R. Aprígio Veloso, 882 – Bodocongó, 58109-900 Campina Grande, PB, Brazil
\textsuperscript{b}Penedo, Campus Arapiraca, Federal University of Alagoas (UFAL), Av. Beira Rio, s/n – Centro Histórico, 57200-000 Penedo, AL, Brazil
\textsuperscript{c}Computing Institute, Federal University of Alagoas (UFAL), Campus A.C. Simões, Cidade Universitária, 57072-970 Maceió, AL, Brazil
\textsuperscript{d}Institute of Mathematics and Computational Sciences, University of São Paulo (USP), Avenida Trabalhador São-carlense, 400 Centro, 13566-590 São Carlos, SP, Brazil

**Article Info**

**Article history:** Available online 25 February 2015

**Keywords:** Feature modeling, Ontology, Software product line, Empirical software engineering

A software product line (SPL) is a set of software systems that have a particular set of common features and that satisfy the needs of a particular market segment or mission. Feature modeling is one of the key activities involved in the design of SPLs. The feature diagram produced in this activity captures the commonalities and variabilities of SPLs. In some complex domains (e.g., ubiquitous computing, autonomic systems and context-aware computing), it is difficult to foresee all functionalities and variabilities a specific SPL may require. Thus, Dynamic Software Product Lines (DSPLs) bind variation points at runtime to adapt to fluctuations in user needs as well as to adapt to changes in the environment. In this context, relying on formal representations of feature models is important to allow them to be automatically analyzed during system execution. Among the mechanisms used for representing and analyzing feature models, description logic (DL) based approaches demand to be better investigated in DSPLs since it provides capabilities, such as automated inconsistency detection, reasoning efficiency, scalability and expressivity. Ontology is the most common way to represent feature models knowledge based on DL reasoners. Previous works conceived ontologies for feature modeling either based on OWL classes and properties or based on OWL individuals. However, considering change or evolution scenarios of feature models, we need to compare whether a class-based or an individual-based feature modeling style is recommended to describe feature models to support SPLs, and especially its capabilities to deal with changes in feature models, as required by DSPLs. In this paper, we conduct a controlled experiment to empirically compare two approaches based on each one of these modeling styles in several changing scenarios (e.g., add/remove mandatory feature, add/remove optional feature and so on). We measure time to perform changes, structural impact of changes (flexibility) and correctness for performing changes in our experiment. Our results indicate that using OWL individuals requires less time to change and is more flexible than using OWL classes and properties. These results provide insightful assumptions towards the definition of an approach relying on reasoning capabilities of ontologies that can effectively support products reconfiguration in the context of DSPL.

© 2015 Elsevier Ltd. All rights reserved.

1. Introduction

Software product line (SPL) engineering is a paradigm that advocates the reusability of software artifacts and the rapid development of new applications for a particular domain (Clements & Northrop, 2001; Pohl, Bockle, & Linden, 2005). These objectives are achieved by capturing the commonalities and variabilities between the products from the same domain in variability models (e.g., feature models). Software product line engineering methods offer characteristics such as rapid product development, reduced time-to-market, quality improvement, and more affordable development costs (Pohl et al., 2005).

Variability management is a fundamental principle in SPL engineering, which involves separating the product line into three main activities: building a common platform, selecting specific requirements for individual products and managing the other two activities throughout SPL development (Hallsteinsen, Hinchey, Park, & Schmid, 2008). Feature modeling (Kang, Cohen,

Hess, Novak, & Peterson, 1990) is broadly used to support variability management both in research and practice. Such kind of modeling is important in the specification of SPLs, since it represents common and variable functionalities of a software family as well as it is used to support the instantiation of applications based on SPL.

Traditional methods for designing SPL mainly focus on its construction in development time (statically), and then later instantiating specific products. However, it is difficult to foresee all functionalities and variabilities an specific SPL may require. In some complex domains (e.g., ubiquitous computing (Kaviani, Mohabbati, Gasevic, & Finke, 2008), autonomic systems (Kephart & Chess, 2003) and context-aware computing (Dargie, 2009)), there’s a need for dynamic SPLs to produce software capable of adapting to fluctuations in user needs and evolving resource constraints. Dynamic Software Product Lines (DSPLs) bind variation points at runtime initially, when software is launched to adapt to the current environment, as well as during operation to adapt to changes in the environment (Hallsteinsen et al., 2008; Hinche, Park, & Schmid, 2012).

To motivate the need to perform changes in feature models in the context of DSPLs, we mention an Ambient Assisted Living (AAL) SPL example in the ubiquitous computing domain. AAL encompasses systems which use different types of sensors, for instance, blood pressure, heart rate, ambient sensors and so on. It monitors activities and vital signs of lonely elderly people to detect emergency situations or deviations from desirable medical patterns (Kleinberger, Becker, Ras, Holzinger, & Müller, 2007). This way, considering a DSPL in this domain, the feature model should take into account these sensors (among other functionalities). Suppose, for example, that a new glucose sensor technology (that monitors blood glucose of a person) is created and it is required to incorporate such sensor in a current product. In this case, as the original feature model does not consider the new glucose sensor technology, changing (evolving) the system and the feature model (e.g., adding the new sensor feature) is required at runtime. Note that the running system can not stop, since it monitors important vital signs of elderly people. In addition, change requests in the product should not have a great impact on the other functionalities of a product.

In order to effectively provide reconfiguration capabilities to feature models, a first step is necessary. According to the study reported by Benavides, Felfernig, Galindo, and Reinfrank (2013), automated analysis of feature models and product configuration have a lot of potential synergies. In this sense, considering the DSPLs context, these two areas are overlapped, since we should have efficient ways to automatic analyze feature models in order to enable products reconfiguration. Thus, relying on formal representations of feature models is important to allow them to be automatically analyzed during system execution.

There are mainly three categories of mechanisms used for the automated analysis of feature models (Benavides et al., 2013): propositional logic based analysis, constraint programming based analysis, and description logic (DL) based analysis. Majority of studies which intend to perform automatic analysis of features use these mechanisms. In addition, as stated by Benavides, Segura, and Ruiz-Cortés (2010), description logic-based methods (e.g., ontologies) demand to be studied in depth to show their strengths and limitations when analyzing feature models. In particular, DL based reasoners may provide a set of potential capabilities to the automated analysis of feature models that could be better investigated, such as automated inconsistency detection, reasoning efficiency, scalability and expressivity (Benavides et al., 2010; Wang, Li, Sunc, Zhang, & Pan, 2007).

Ontology is the most common way to represent feature models knowledge based on DL reasoners. Ontologies languages are used for domain formalization by defining classes and properties for these classes, individuals (that instantiate the classes), properties of individuals, and statements on these individuals. It also allows to reason about these classes and individuals according to formal semantics defined by the language. Ontologies are generally represented using one of the variants of the Web Ontology Language (OWL) (McGuinness & van Harmelen, 2015) which is part of the technologies stack defined by the World Wide Web Consortium (W3C) for Semantic Web. In this way, previous works use OWL to formally represent feature diagrams, aiming to provide means for automatic reasoning. These works conceived ontologies for feature modeling either based on OWL classes and properties (e.g., Wang et al., 2007) or based on OWL individuals, for example the approaches proposed by Zaid, Kleinermann, and De Troyer (2009) and Tenório, Dermeval, and Bittencourt (2014).

However, considering change or evolution scenarios of feature models – for instance the case of complex domains that require dynamic SPLs – we argue that the way on which ontologies are modeled may significatively impact on the flexibility to reason and/or to modify ontologies for feature modeling. In this way, we need to compare whether a class-based or an individual-based feature modeling style is recommended to describe feature models to support SPLs, and especially to deal with changes in feature models, as required by DSPLs.

In this paper, we conduct a controlled experiment that compares the ontology proposed by Wang et al. (2007) (based on OWL classes and properties) and the one proposed by Tenório et al. (2014) (based on instances/individuals) in several changing scenarios 1 (e.g., add/remove mandatory feature, add/remove optional feature, add/remove alternative feature and so on). Our empirical comparison takes into account metrics such as, time to perform a change, structural impact of a change and correctness for performing a change. We design and execute it and then we statistically analyze data gathered from the experiment. It is executed with ten participants in an academic context.

The results obtained with this empirical comparison provide insightful assumptions to help identifying which ontology-based feature modeling style is more suitable regarding its capabilities to handle change or evolution requests in feature models. These assumptions are of outmost importance towards the definition of an approach relying on reasoning capabilities of ontologies that can effectively support both automatic analysis of feature models and products reconfiguration, in the context of DSPL.

The remainder of this paper is organized as follows. Section 2 presents the main concepts of feature modeling and gives an overview of the two ontologies compared in this study. Section 3 describes in detail the design of this study’s experiment. Section 4 depicts how the experiment was executed. Section 5 analyzes the data collected in the experiment. Section 6 summarizes our results, pointing out open issues and related works, and also discusses any threats to the validity of our work. Finally, Section 7 presents conclusions and future works.

2. Background

This section presents feature modeling concepts and definitions, along with a graphical notation. It also illustrates the two ontologies for representing feature models, which are compared in the experiment.

1 Note that we chose the ontology proposed by Tenório et al. (2014) because that ontology was explicitly designed to support dynamic changes in the feature models. Moreover, we chose the approach proposed by Wang et al. (2007) because it is one of the first approaches that use ontology in feature modeling and it is also published in a high reputation venue on Semantic Web field.
2.1. Feature modeling

The variability of SPLs is commonly expressed through features represented in feature models. A feature is a property of the system that is relevant to some stakeholder and is used to capture similarities and variabilities of software systems. Feature modeling has been proposed as an approach for describing variable requirements for software product lines (Czarnecki, Peter Kim, & Kalleberg, 2006). It is an important activity of the software product line development process, since it is in such phase that the common and variable features of the product family are specified.

Features are organized in feature models according to one of the following types: mandatory, optional, alternative, and or-features. The mandatory type must be present in all products derived from a software product line. The optional one may or may not be included in a product derived from a SPL, hence its presence is optional. In the alternative feature, exactly one feature from a set of features must be included in a product. In the or-feature type, one or more features from a set of features can be included in a product from a SPL.

The most widely used technique for defining features was originally presented by Kang et al. (1990), named Feature-Oriented Domain Analysis (FODA). FODA provides a graphical tree-like notation that shows the hierarchical organization of features. The root of the tree represents the whole SPL node and all other nodes represent different types of features that are part of a SPL.

Fig. 1 presents the feature model of a smartphone SPL, represented in FODA notation. This feature model was adapted from a repository of feature models² and is used as the experimental unit of this work, further detailed in Section 3.

The variability of SPLs is commonly expressed through features represented in feature models. Features are organized in feature models according to one of the following types: mandatory, optional, alternative, and or-features. Mandatory features are graphically represented by a small, filled black circle above the feature name (e.g., Operational System, Call and Screen). Optional features are graphically specified by an open, non-filled white circle (e.g., GPS, Flash and Media). Alternative features share the same parent feature and are graphically represented by an open arc situated just below the parent feature (e.g., Android, iOS and Windows Phone, and Basic, Color and High Resolution). Finally, the or-features (e.g., Camera, MP3 and Radio) are represented by a filled arc, similar to the alternative features. The black box on below the feature model depicts a legend for the feature model elements.

Additionally, in the feature modeling using FODA notation, it is possible to represent dependency rules between features, which can be one of two types: (i) Requires, when one feature requires the existence of another feature (they are interdependent), and (ii) Excludes, when one feature is mutually exclusive to another one (they can not coexist).

2.2. Ontology-based feature modeling

Ontology can be defined as an “explicit specification of a conceptualization” (Gruber, 1993). It is explicit because of its classes and properties visibility. Conceptualization is understood to be an abstract and simplified version of the world to be represented. Moreover, ontologies are formal because they can be logically reasoned and are also shared because they should be agreed by actors within a specific domain (Guarino, 1998). Ontology has its roots in description logics (Baader et al., 2003) and is a standard form for representing the concepts within a domain, as well as the relationships between those concepts in a way that allows automated reasoning. It is generally represented using one of the variants of the Web Ontology Language (OWL) (Antoniou et al., 2004).

In this section two ontology-based feature modeling styles are presented. First, we present the ontology proposed by Wang et al. and then we present the ontology proposed by Tenório et al.

2.2.1. Ontology-based feature modeling by Wang et al. (2007)

Hereafter, we present how a feature diagram and additional constraints are modeled in OWL according to Wang et al. (2007). First, the feature modeling with OWL is explained and later each one of the feature types is described.

The ontology is constructed in a number of steps:

- Step 1. Identify the nodes (concepts and features) present in a feature diagram. Each node in the diagram is modeled as an OWL class.
- Step 2. For each of these nodes in the diagram, a rule class is created. This rule class has two kinds of conditions: firstly, a necessary and sufficient (NS.EquivalentClass) condition using an existential restriction to bind the rule node to the corresponding feature node in the diagram; secondly, a number of (possibly 0) necessary (N.subClassOf) constraints later, serving two purposes: to specify how each of its child features are related to this node, capturing the various relations between features and to specify how this feature node is constrained by other features, in the form of requires and excludes.
- Step 3. The root concept and features in a feature diagram are inter-related by various feature relations, represented by different edge types in the diagram. For each of these edges, an object-property is created. It is asserted that the range of the property is the respective feature class.

All axioms of the ontologies described in this paper are formally defined in description logics. The OWL syntax used to represent such axioms is summarized in Table 1.

In general terms, for a parent feature \( G \) and its child features \( F_1,...,F_n \), the initial modeling above produces the following ontology.

\[
\begin{align*}
G & \sqsubseteq T \\
GRule & \sqsubseteq T \\
G & \sqsubseteq \exists hasG.G \\
GRule & \equiv \exists hasG.G \\
F_1 & \sqsubseteq T \\
F_1Rule & \sqsubseteq T \\
F_1 & \sqsubseteq \exists hasF_1.F_1 \\
F_1Rule & \equiv \exists hasF_1.F_1 \\
\vdots & \\
F_n & \sqsubseteq T \\
F_nRule & \sqsubseteq T \\
F_n & \sqsubseteq \exists hasF_n.F_n \\
F_nRule & \equiv \exists hasF_n.F_n \\
G & \sqsubseteq \neg F_i,F_j for1 \leq i \leq n \\
F_i & \sqsubseteq \neg F_j,F_j for1 \leq i,j \leq n \land i \neq j
\end{align*}
\]

In following, the feature relations using the ontology are defined. The general definition of each of the four feature relations are shown, based on the above feature ontology.

- **Mandatory.** For each of the mandatory features \( F_1,...,F_n \) of a parent feature \( G \), each \( N \) constraint in \( GRule \) is used to model it. It is a someValuesFrom restriction on \( hasF_n \), stating that each instance of the rule class must have some instance of \( F_1 \) class for \( hasF_n \). The following ontology fragment shows the modeling of mandatory feature set and parent feature \( G \).

\[
\begin{align*}
GRule & \sqsubseteq \exists hasF_1.F_1 \\
\vdots & \\
GRule & \sqsubseteq \exists hasF_n.F_n
\end{align*}
\]
Optional. According to the feature modeling of Wang et al. (2007), for each of the optional features $F_1, \ldots, F_n$ of a parent feature $G$, no additional statements are required to model this relationship.

Alternative. For a set of alternative features $F_1, \ldots, F_n$ and a parent feature $G$, a disjunction of someValuesFrom restrictions is used over hasFi to ensure that some feature will be included. The complement of the distributed disjunction of the conjunction of two someValuesFrom restrictions is used to ensure that only one feature can be included. The symbol $\cup$ represents distributed disjunction.

$$\text{GRule} \subseteq \bigcup_{1 \leq i \leq n} \text{hasFi}, F_i$$

$$\text{GRule} \subseteq \neg \bigcup_{1 \leq i \leq j \leq n} \text{hasFi} \cap \text{hasFj}, F_j$$

Or-feature. For a set of or-features $F_1, \ldots, F_n$ of a parent feature $G$, a disjunction of someValuesFrom restrictions must be used to model the relation.

$$\text{GRule} \subseteq \bigcup_{1 \leq i \leq n} \text{hasFi}, F_i$$

One may notice that the definition of or-feature is very similar to the alternative feature, with the omission of the negation of distributed disjunction to allow for multiple or-features to be included.

Requires. For a given feature $G$ and a set of features $F_1, \ldots, F_n$ that $G$ requires, besides the NS condition that binds GRule to $G$, it is certain that each of the $F_i$ features appears in a configuration if $G$ is present.

$$\text{GRule} \subseteq \exists \text{hasFi}, F_i$$

$$\text{GRule} \subseteq \exists \text{hasFj}, F_j$$

Excludes. For a given feature $G$ and a set of features $F_1, \ldots, F_n$ that $G$ excludes, it is certain, using the negation of someValues – From restriction on hasFi property, that GRule does not have any $F_i$ feature.

$$\text{GRule} \subseteq \neg \exists \text{hasFi}, F_i$$

$$\text{GRule} \subseteq \neg \exists \text{hasFj}, F_j$$

2.2.2. Ontology-based feature modeling by Tenório et al. (2014)

In this section, the ontology proposed by Tenório et al. is described. Its concepts, properties and relationships are presented, along with the definition of the ontology in terms of axioms, i.e., description logics.

Hereafter, the classes, properties and concepts of this ontology are presented:

- **SoftwareProductLine** (name, description, FeatureModel): this class represents an arbitrary software product line. It has primitive elements such as name and description. Moreover, a SPL contains a Feature model.
- **FeatureModel** (name, Feature, FeatureConstraint): this class describes a Feature Model that represents the hierarchical organization of the features of an SPL. It has a set of features and a set of feature constraints.
- **Feature** (name): this class represents a resource available in the software product line. It may be classified into Mandatory, Optional or Alternative:
  - **Mandatory** (name): this class represents a mandatory resource of the SPL, i.e., it must be present in all products.
– **Optional (name)**: this class represents an optional resource of the SPL, i.e., it is optionally present in any product.
– **Alternative (name, exclusive, AlternativeFeature)**: this class represents an alternative resource of the SPL. An alternative resource specifies that two or more resources may not co-exist.

- **FeatureConstraint (name)**: this class represents a constraint in the feature model. It may be classified into Depend, Exclude or Group:

  – **Depend (name, SourceFeature, TargetFeature)**: this class represents a constraint of the Depend type. As mentioned above, it has a set of source features and a set of target features.
  – **Exclude (name, SourceFeature, TargetFeature)**: this class represents a constraint of the Exclude type. As mentioned above, it has a set of source features and a set of target features.
  – **Group (name, SetFeatures, typeConstraint)**: this class represents a constraint of the Group type. It has a set of features and a typeConstraint that indicates the type of the constraint. It can be: (i) zero-or-one feature exactly (0 or 1). (ii) At-least-one feature (1 or more), (iii) Exactly-one feature (1), (iv) Any feature (0 or more), or (v) All features (n).

The following relationships are represented in the ontology:

- **hasRootFeatures (FeatureModel, Feature)**: specifies that a FeatureModel contains a set of root features (which may not be empty).
- **hasSetOfAlternativeFeatures (Alternative, Alternative)**: specifies that an alternative feature must have at least one feature alternative. It is a symmetric property.
- **hasSetOfConstraints (FeatureModel, FeatureConstraint)**: specifies that a FeatureModel contains a set of feature constraints.
- **hasSetOfFeatures (Group, Feature)**: specifies that a Group constraint contains a set of features (which may not be empty).
- **hasSourceFeatures (Depend/Exclude, Feature)**: specifies that a Depend or Exclude constraint has a set of source features (which must have at least one feature).
- **hasTargetFeatures (Depend/Exclude, Feature)**: specifies that a Depend or Exclude constraint has a set of target features (which must have at least one feature).
- **isBasedOn (SoftwareProductLine, FeatureModel)**: specifies that a SPL is based on exactly one FeatureModel. It is a functional property.
- **isChildOf (Feature, Feature)**: specifies that a feature is the child of exactly one another feature. It is a functional property and it is also the inverse property of isParentOf.
- **isParentOf (Feature, Feature)**: specifies that a feature contains a set of children features. It is the inverse property of isChildOf.

Moreover, the classes, object properties and data properties axioms are presented in Tables 2–4, respectively.

### Table 2

<table>
<thead>
<tr>
<th>Classes axioms of the ontology.</th>
<th>Axioms</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Class</strong></td>
<td><strong>Axioms</strong></td>
</tr>
<tr>
<td>Alternative</td>
<td>Alternative ⊑ Feature</td>
</tr>
<tr>
<td>Optional</td>
<td>Alternative ⊑ Optional</td>
</tr>
<tr>
<td>HasSetOfAlternativeFeatures</td>
<td>HasSetOfAlternativeFeatures ⊑ Alternative</td>
</tr>
<tr>
<td>HasSetOfConstraints</td>
<td>HasSetOfConstraints ⊑ FeatureModel</td>
</tr>
<tr>
<td>HasSetOfFeatures</td>
<td>HasSetOfFeatures ⊑ Group</td>
</tr>
<tr>
<td>HasRootFeatures</td>
<td>HasRootFeatures ⊑ FeatureModel</td>
</tr>
<tr>
<td>HasTargetFeatures</td>
<td>HasTargetFeatures ⊑ Group</td>
</tr>
<tr>
<td>IsBasedOn</td>
<td>IsBasedOn ⊑ SoftwareProductLine</td>
</tr>
<tr>
<td>IsChildOf</td>
<td>IsChildOf ⊑ Feature</td>
</tr>
<tr>
<td>IsParentOf</td>
<td>IsParentOf ⊑ FeatureModel</td>
</tr>
</tbody>
</table>

3. **Experimental design**

This section explains in detail the experiment planning that took place for this work. Following this, the research hypotheses of our experiment, the independent and dependent variables, the experimental unit that is applied to the treatments and the selection of the experiment design is explained.

The empirical strategy adopted in this work is a controlled experiment of a narrow comparison type. This experiment is applied in an academic context with a set of participants; more details about the experiment is further explained.

3.1. **Research hypotheses**

Before defining the research hypotheses of the experiment, the research question that guided the specification of such hypotheses is presented below:

**Research Question** – are there significative differences in the use of two different ontology-based feature modeling styles (based on OWL classes and based on OWL individuals) regarding different scenarios of change? If yes, which ontology is more suited to be used in the context of dynamic software product lines?.

The above research question suggest the analysis of both ontologies with regards to different metrics. Firstly, as flexibility (impact of change) is a fundamental characteristic that features models which are constantly changing/evolving should deal, such metric must be empirical analyzed in both ontologies being compared. In this experiment, a set of participants perform different kinds of changes in a reference feature model (as will be further explained), thus the correctness of such changes should also be analyzed to verify if the changes are correctly performed and also to verify the understanding of ontologies modeling by the participants. Moreover, monitoring how long time the participants take to perform the different scenarios of change is also important. It is used to estimate the cost for realizing specific scenarios of change in the feature models represented by the both ontologies compared in this experiment. The importance of such metrics for investigating the above research question lead to the following research hypotheses of this experiment:
Data properties axioms of the ontology.

<table>
<thead>
<tr>
<th>Data property</th>
<th>Axioms</th>
</tr>
</thead>
<tbody>
<tr>
<td>description</td>
<td>⊑ C 1description</td>
</tr>
<tr>
<td>exclusive</td>
<td>⊑ C 1exclusive</td>
</tr>
<tr>
<td>name</td>
<td>⊑ C 1name</td>
</tr>
<tr>
<td>typeConstraint</td>
<td>⊑ C 1typeConstraint</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Object property</th>
<th>Axioms</th>
</tr>
</thead>
<tbody>
<tr>
<td>hasRootFeatures</td>
<td>⊑ C FeatureModel</td>
</tr>
<tr>
<td></td>
<td>⊑ C VhasRootFeatures(⊑ C hasRootFeaturesThing ⊑ C FeatureModel)</td>
</tr>
<tr>
<td>hasSetOfAlternativeFeatures</td>
<td>⊑ C hasSetOfAlternativeFeatures = hasSetOfAlternativeFeatures</td>
</tr>
<tr>
<td></td>
<td>⊑ C VhasSetOfAlternativeFeatures(⊑ C Alternative)</td>
</tr>
<tr>
<td></td>
<td>⊑ C VhasSetOfAlternativeFeatures(⊑ C Alternative)</td>
</tr>
<tr>
<td>hasSetOfConstraints</td>
<td>⊑ C hasSetOfConstraintsThing ⊑ C FeatureModel</td>
</tr>
<tr>
<td></td>
<td>⊑ C VhasSetOfConstraints(⊑ C hasSetOfConstraints Thing)</td>
</tr>
<tr>
<td></td>
<td>(Depend ⊑ C Exclude ⊑ C Group)</td>
</tr>
<tr>
<td>hasSetOfFeatures</td>
<td>⊑ C hasSetOfFeaturesThing ⊑ C Group</td>
</tr>
<tr>
<td></td>
<td>⊑ C VhasSetOfFeatures(⊑ C hasSetOfFeaturesFeature)</td>
</tr>
<tr>
<td>hasSourceFeatures</td>
<td>⊑ C hasSourceFeaturesThing ⊑ C Depend</td>
</tr>
<tr>
<td></td>
<td>⊑ C VhasSourceFeatures(⊑ C to Dependency)</td>
</tr>
<tr>
<td></td>
<td>⊑ C VhasSourceFeatures(⊑ C Exclude)</td>
</tr>
<tr>
<td></td>
<td>⊑ C VhasSourceFeatures(⊑ C Alternative)</td>
</tr>
<tr>
<td>hasTargetFeatures</td>
<td>⊑ C hasTargetFeaturesThing ⊑ C Depend</td>
</tr>
<tr>
<td></td>
<td>⊑ C VhasTargetFeatures(⊑ C Alternative)</td>
</tr>
<tr>
<td></td>
<td>⊑ C VhasTargetFeatures(⊑ C Exclude)</td>
</tr>
<tr>
<td></td>
<td>⊑ C VhasTargetFeatures(⊑ C to Dependency)</td>
</tr>
<tr>
<td>isBasedOn</td>
<td>⊑ C 1isBasedOnThing</td>
</tr>
<tr>
<td></td>
<td>⊑ C VisBasedOn(⊑ C SoftwareProductLine)</td>
</tr>
<tr>
<td>isChildOf</td>
<td>⊑ C 1isChildOfFeature</td>
</tr>
<tr>
<td></td>
<td>⊑ C VhasChildOf(⊑ C Feature)</td>
</tr>
<tr>
<td></td>
<td>⊑ C VhasChildOf(⊑ C to Dependency)</td>
</tr>
<tr>
<td></td>
<td>(isParentOf ⊑ C Feature)</td>
</tr>
<tr>
<td></td>
<td>⊑ C isParentOf ⊑ C Feature</td>
</tr>
<tr>
<td></td>
<td>⊑ C VhasParentOf(⊑ C Feature)</td>
</tr>
<tr>
<td></td>
<td>⊑ C VhasParentOf(⊑ C to Dependency)</td>
</tr>
</tbody>
</table>

**H1** = 0 : The time for performing a change in the ontologies is equal

**H1** = 1 : The time for performing a change in the ontologies is different

**H2** = 0 : The structural impact of a change in the ontologies is equal

**H2** = 1 : The structural impact of a change in the ontologies is different

**H3** = 0 : The correctness for performing a change in the ontologies is equal

**H3** = 1 : The correctness for performing a change in the ontologies is different

In Table 5 these research hypotheses are formally presented. \( T, I \) and \( C \) are functions that return, respectively, the value of the time for change, the impact of change and the correctness of change metrics on the ontologies \( O1 \) (Wang et al., 2007) and \( O2 \) (Tenório et al., 2014).

### 3.2. Variables selection

In this section, we define the variables of the experiment. First, we explain the independent variables, also named factors, and next, we depict the dependent variables.

The independent variables of the experiment are defined as follows. Furthermore, the factor levels are defined according to Table 6.

- Change scenarios: this variable refers to 14 change tasks that are applied to the ontologies, representing the feature model of the Smartphone SPL (see Fig. 1). There are scenarios of two change types: corrective and evolutionary.
- Ontologies: this variable specifies the ontology on which the change scenarios are applied.

The dependent variables (also named metrics) are defined in more detail below:

- Time for changing \( T \): this variable captures the time interval required to perform a change task.
- Impact of change \( I \): this variable captures the structural impact of a change. In terms of ontologies, this metric is measured through the total number of classes and properties of a ontology. This metric is inversely related to flexibility, i.e., the greater the change impact, the less the flexibility. Eq. (1) defines how this metric is calculated.

\[
I = \text{number Of Classes} + \text{number Of Properties} \tag{1}
\]
Correctness of change (C): this variable sets the correctness level of a change performed in an ontology. This metric is a ratio between the number of correct steps performed to make a change and the total number of correct steps that should have been performed, as stated in Eq. (2). Tables 7 and 8 present the set of steps required to correctly perform all the fourteen scenarios of change in both ontologies.

\[
C = \frac{\text{number Of Correct Steps}}{\text{total Of Correct Steps}}
\]

3.3. Experimental units

As explained in Section 2, the experimental unit of this empirical study is the Smartphone SPL (see feature model in Fig. 1).

3.4. Experimental design selection

Considering the design of experiment classification (Juristo & Moreno, 2010), this experiment is classified as a full factorial design with repetitions. Each repetition is applied to a different participant. This way, since the experiment has two factors, with 14 and 2 levels, the total number of trials is 28. Table 9 depicts each one of the treatments applied to the experimental unit.

Note that this experiment is balanced since, for each repetition, the same number of treatments is applied. Moreover, the sequence of application of the treatments is randomly allocated by ontology, i.e., one participant first uses Ontology 1 and then Ontology 2, while the other participant first uses Ontology 2 and then Ontology 1 and so on. Regarding the changing scenarios, the order of application follows the sequence defined in Table 6.

4. Experiment execution

This section describes how the experiment was executed. It depicts who the participants are (and how they were selected), which instruments were used and how the experiment was performed.

4.1. Participant selection

The experiment involves the active participation of human agents. Five undergraduate students and five master students in computer science or computer engineering were selected. These students were involved in the same research group, were enrolled in different periods and had prior knowledge on ontology and experience with the Protégé tool. All students were affiliated to the Federal University of Alagoas (Brazil), the location where the experiment was conducted.

4.2. Preparation and instrumentation

We followed a set of steps to properly start the experiment:

1. Specification of the Smartphone SPL feature model (based on a set of similar feature models) in the two ontologies. Fourteen copies of Ontology 1 (.OWL file) and 14 copies of Ontology 2 (.OWL file) were provided to each participant to contain each of the change scenarios.

2. Specification of the procedure for data collection during the experiment. A web system (available at <http://nees.com.br/ontoFeatureModelsExperiment/en/>) was developed to record the time taken by the participants to perform the tasks of changes.

3.3. Experimental units

As explained in Section 2, the experimental unit of this empirical study is the Smartphone SPL (see feature model in Fig. 1).

3.4. Experimental design selection

Considering the design of experiment classification (Juristo & Moreno, 2010), this experiment is classified as a full factorial design with repetitions. Each repetition is applied to a different participant. This way, since the experiment has two factors, with 14 and 2 levels, the total number of trials is 28. Table 9 depicts each one of the treatments applied to the experimental unit.

Note that this experiment is balanced since, for each repetition, the same number of treatments is applied. Moreover, the sequence of application of the treatments is randomly allocated by ontology, i.e., one participant first uses Ontology 1 and then Ontology 2, while the other participant first uses Ontology 2 and then Ontology 1 and so on. Regarding the changing scenarios, the order of application follows the sequence defined in Table 6.

4. Experiment execution

This section describes how the experiment was executed. It depicts who the participants are (and how they were selected), which instruments were used and how the experiment was performed.

### Table 6

Factors levels.

<table>
<thead>
<tr>
<th>Factor</th>
<th>Levels</th>
</tr>
</thead>
<tbody>
<tr>
<td>Change scenario</td>
<td>C1 – Remove Call Mandatory Feature</td>
</tr>
<tr>
<td></td>
<td>C2 – Add SMS Mandatory Feature</td>
</tr>
<tr>
<td></td>
<td>C3 – Remove Flash Optional Feature</td>
</tr>
<tr>
<td></td>
<td>C4 – Add Bluetooth Optional Feature</td>
</tr>
<tr>
<td></td>
<td>C5 – Remove Operational System Alternative Feature and its variants</td>
</tr>
<tr>
<td></td>
<td>C6 – Add Input Alternative Feature (Mandatory) and its variants</td>
</tr>
<tr>
<td></td>
<td>C7 – Remove Media Or-Feature and its variants: Camera, MP3 and Radio</td>
</tr>
<tr>
<td></td>
<td>C8 – Add Input Or-Feature and its variants: Touch, Keypad and Speech</td>
</tr>
<tr>
<td></td>
<td>C9 – Remove Requires Constraint (High Resolution — Camera)</td>
</tr>
<tr>
<td></td>
<td>C10 – Add Requires Constraint (Flash — Camera)</td>
</tr>
<tr>
<td></td>
<td>C11 – Remove Excludes Constraint (GPS — Basic)</td>
</tr>
<tr>
<td></td>
<td>C12 – Add Excludes Constraint ( — Basic)</td>
</tr>
<tr>
<td></td>
<td>C13 – Remove Windows Phone from Operational System Alternative</td>
</tr>
<tr>
<td></td>
<td>C14 – Add BlackBerry OS variant to the Operational System Alternative</td>
</tr>
<tr>
<td>Ontology</td>
<td>O1 – Ontology by Wang et al. (2007)</td>
</tr>
<tr>
<td></td>
<td>O2 – Ontology by Tenório et al. (2014)</td>
</tr>
</tbody>
</table>

### Table 7

Steps to perform correct changes in Ontology 1.

<table>
<thead>
<tr>
<th>Change scenario</th>
<th>Correct steps</th>
<th>Number of steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>1. Remove 2 classes</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>2. Remove 1 object property</td>
<td></td>
</tr>
<tr>
<td>C2</td>
<td>1. Create 2 classes</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>2. Create equivalent to axiom</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3. Create 1 object property</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4. Set subclassof constraint</td>
<td></td>
</tr>
<tr>
<td>C3</td>
<td>1. Remove 2 classes</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>2. Remove 1 object property</td>
<td></td>
</tr>
<tr>
<td>C4</td>
<td>1. Create 2 classes</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>2. Create equivalent to axiom</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3. Create 1 object property</td>
<td></td>
</tr>
<tr>
<td>C5</td>
<td>1. Remove 8 classes</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>2. Remove 4 object properties</td>
<td></td>
</tr>
<tr>
<td>C6</td>
<td>1. Create 8 classes</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td>2. Create 4 object properties</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3. Create 4 equivalent to axioms</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4. Create 2 subclassOf constraints (alternative)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5. Create 1 subclassOf constraint (in input feature)</td>
<td></td>
</tr>
<tr>
<td>C7</td>
<td>1. Remove 8 classes</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>2. Remove 4 object properties</td>
<td></td>
</tr>
<tr>
<td>C8</td>
<td>1. Create 8 classes</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>2. Create 4 object properties</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3. Create 4 equivalent to axioms</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4. Create 1 subclassOf constraint (or-feature)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5. Create 1 subclassOf constraint (in input feature)</td>
<td></td>
</tr>
<tr>
<td>C9</td>
<td>1. Remove 1 constraint</td>
<td>1</td>
</tr>
<tr>
<td>C10</td>
<td>1. Add constraint</td>
<td>1</td>
</tr>
<tr>
<td>C11</td>
<td>1. Remove 1 constraint</td>
<td>1</td>
</tr>
<tr>
<td>C12</td>
<td>1. Add constraint</td>
<td>1</td>
</tr>
<tr>
<td>C13</td>
<td>1. Remove 2 classes</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>2. Remove 1 object property</td>
<td></td>
</tr>
<tr>
<td>C14</td>
<td>1. Create 2 classes</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>2. Create 1 object property</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3. Create equivalent to axiom</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4. Update 2 constraints</td>
<td></td>
</tr>
</tbody>
</table>
3. Elaboration of the instructional material used to prepare the participants for the experiment. It includes guidelines on how to specify each type of change in both ontologies.

The necessary instruments to be used in the experiment were:
(i) a PC with a reasonable configuration for each one of the participants; (ii) the Protégé 4.3 tool installed on the PCs, to be used as an OWL tool on which all the tasks changes would be performed; and (iii) access to Dropbox software for the participants to save the ontologies (.OWL files) for each change performed.

4.3. Narrative of the experiment

The execution steps of the experiment are sequentially described as follows:

1. Before the experiment, the participants received the necessary instructions about their activities in the experiment.

2. All participants received the set of tasks to be performed, along with the order of performing such changes in the ontologies. The order of ontologies for each one of the participants can be seen at <http://nees.com.br/ontoFeatureModelsExperiment/en/>.

3. After the participants finished their tasks, all the .OWL files regarding the change scenarios of both ontologies were gathered.

4. The results of the three metrics were organized in a.txt file. The time interval was directly captured by the web system for collecting data, whereas the other metrics (impact and correctness) were obtained by analyzing the .OWL files modified by the participants. The impact metric was calculated with the aid of the Pellet reasoner (Sirin, Parsia, Grau, Kalyanpur, & Katz, 2007).

5. After data collection, data were analyzed with the aim of answering the research hypotheses. In the following section the analysis of the experiment is fully described.

5. Experiment analysis

This section presents the analysis of the data collected in the experiment execution. The collected data as well as the scripts and spreadsheets used in the experimental analysis can be accessed at: <http://nees.com.br/ontoFeatureModelsExperiment/ExperimentData.rar>.

The data contain 280 instances corresponding to each combination of participant, scenario and ontology. Moreover, data also include the following attributes: time, structural_impact and correctness. Time is recorded in milliseconds. The structural_impact attribute records a natural number indicating the number of OWL classes and properties and the correctness attribute records a proportion number of correctness from 0 (less correct) to 1 (totally correct).

Following the experimental analysis process proposed by Juristo and Moreno (2010), the experimental analysis includes for each of the dependent variables: descriptive analysis of the data,
identification of the mathematical model, validation of the mathematical model and analysis of variation of the effects. These steps are depicted in Fig. 2.

5.1. Descriptive analysis

The descriptive analysis mainly encompasses the graphical presentation of boxplots (comparing the results of the two ontologies) and the summarization of the statistics of each metric. With this analysis it is possible to obtain preliminary indications of answers to the raised research hypotheses and it also allows a better understanding of the metrics (time for change, impact of change and correctness of change) behavior.

5.1.1. Time for changing

Fig. 3 presents the boxplots of the time for performing changes by each one of the ontologies. It suggests that the time for changing Ontology 1 (Wang et al.) is higher than for changing Ontology 2 (Tenório et al.). However, at this time there is no statistical evidence to support this; such conclusions are only statistically significant when statistical tests are performed, as will be explained in Section 5.4. Table 10 presents the summary of statistics of the time for performing change data in the two ontologies.

5.1.2. Impact of change

Fig. 4 presents the boxplots of the structural impact for performing changes for each one of the ontologies. It suggests that the impact of change in Ontology 1 (Wang et al., 2007) is higher than the impact of change in Ontology 2 (Tenório et al., 2014). In fact, it was expected that the impact of change in Ontology 1 would be higher than the impact of change in Ontology 2, since the first ontology is based on classes, which characterizes a higher structural impact in the ontology. For Ontology 2, any task of change impacts in the ontology structure, since all changes are made at the level of instances; hence, there is no variance in the data of this ontology. Table 11 presents the summary of statistics of the impact of change data for the two ontologies.

5.1.3. Correctness of change

Fig. 5 presents the boxplots of the correctness of changes by each one of the ontologies. These graphics do not allow us to draw any conclusions, since the values of both ontologies are very close. Table 12 presents the summary of statistics of the correctness of change data for the two ontologies.

5.2. Mathematical model identification

The theoretical model of the experimental design selected (full factorial with 2 factors and 10 repetitions) considers two main effects (\(a\) and \(b\)) and an interaction between the two factors (\(ab\)). This model is expressed by Eq. (3) (Juristo & Moreno, 2010), where \(Y_{ijk}\) is the estimated value of the dependent variable for the ontology \(i\), task of change \(j\) and repetition \(k\); \(\mu\) is the estimated mean of the dependent variable considering all observations; \(\alpha_i\) is the estimated effect of the ontology \(i\) on the output; \(\beta_j\) is the estimated effect of the task of change \(j\) on the output; \(\alpha\beta_{ij}\) is the interaction between the primary factors and \(\epsilon_{ijk}\) is the experimental error of the ontology \(i\), the task change \(j\) and repetition \(k\).

\[
Y_{ijk} = \mu + \alpha_i + \beta_j + \alpha\beta_{ij} + \epsilon_{ijk}
\] (3)

Since this experiment captures the data regarding three dependent variables, in the following sections the mathematical model of each one of these variables is presented.
5.2.1. Time for changing mathematical model

The identified model (through data analysis) of the time ($T_{ij}$) for changing the ontologies metric is represented by Eq. (4). As will be further explained, only the effect of the ontology factor ($\alpha_i$) and the effect of the interaction between the factors ($\alpha_i\beta_j$) are statistically significant to this variable. Thus, with the aim of simplification, the time for changing model contains only the coefficients of the ontology factor and of the interaction between factors. The number 169885.3 is the estimated mean of the model.

$$T_{ij} \approx 169885.3 + \alpha_i + \alpha_i\beta_j \quad (4)$$

5.2.2. Impact of change mathematical model

The identified model of the impact ($I_{ij}$) of change in the ontologies is represented by Eq. (5). For this metric, the effect of all factors and of the interaction between them are statistically significant (as seen in Section 5.4). The estimated mean of the model is 36.2.

$$I_{ij} \approx 36.2 + \alpha_i + \beta_j + \alpha_i\beta_j \quad (5)$$

5.2.3. Correctness of change mathematical model

The model of the correctness of change metric is represented by Eq. (6). In this model, only the ontology factor is statistically significant (as we will see in Section 5.4); thus, for the sake of simplicity, the other effects are removed from the model. The estimated mean of the model is 0.793795.

$$C_{ij} \approx 0.793795 + \alpha_i \quad (6)$$

5.3. Model validation

Since the mathematical model of the metrics is identified, it is possible to validate the metrics and then statistically test the experiment hypotheses using the ANOVA (Analysis of Variance) method (Festing, 2015). The identified models are evaluated through a residual analysis. Thus, the assumptions of (a) normality, (b) independence of errors, and (c) constant variation of the residuals were verified for each one of the dependent variables. Such analysis is prerequisite for the ANOVA test execution (Juristo & Moreno, 2010).

5.3.1. Validation of the time for changing model

As seen in Fig. 6, the residuals of this variable are not normal. Analysis of the dispersion graphics presented in the same figure shows that the errors are independent, since there is no strong correlation between the errors. Moreover, it is not possible to identify standards of continuous growth (funnel) in the dispersion graphics between time for changing and residuals, hence there is no evidence that the variation is not constant.

5.3.2. Validation of the impact of change model

Fig. 7 and the execution of a Shapiro–Wilk test demonstrate that the residuals of the impact of change metric are not normal. Besides that, similarly to the time for changing model, analysis of the dispersion graphics of this variable shows that the errors are independent. It is also impossible to identify standards of continuous growth in the dispersion graphics, thus there is no evidence that the variation is not constant.

5.3.3. Validation of the correctness of change model

Similarly to the previous metrics, Fig. 8 illustrates that the residuals of the correctness of change metric are not normal. Furthermore, by observing the dispersion graphics of the same figure, we can see that the errors are independent and that there is no evidence to state that the variation is not constant.

According to the analysis of residuals of the three metrics above, it can be stated that the models are adequate to the data.
hypotheses are verified using confidence intervals and statistical tests.

5.4.1. Allocation of variation
As shown in Table 13, considering all the effects, the variation on time for changing and impact of change metrics is mostly due to the effects. Regarding the time for changing variable, the effects are responsible for almost 70% of the total variation (despite the minimal influence of the β effect). For this metric, the ontology effect is the most important one.

Similarly, the effects of factors are responsible for the largest portion (98.81%) of the variation of the impact of change variable. Furthermore, the most important factor for this metric is the effect of the scenario of change factor.

Concerning the correctness metric, Table 13 depicts that there is a very significant part of the total variation that is assigned to the experimental errors (95.42%). In this case, it is not possible to explain the variation on the output with only the two factors considered in the experiment. It is possible that some important factor regarding this metric has not been observed (e.g., participants experience with the Protégé tool, understanding of the ontology and so on). Moreover, note that the ontology factor is the most important factor (3.3%) with respect to the correctness metric.

5.4.2. Effects signification
For each factor and interaction between factors – in the three metrics – of the experiment, the following hypotheses are verified:

Hn – 0 : There is no difference between alternatives effects of the factor in the dependent variable
Hn – 1 : There is difference between alternatives effects of the factor in the dependent variable

With respect to the time for changing metric, performing a F-Test with 0.05 of significance level, the null hypothesis is rejected (F-Table < F value) for the ontology factor and for the interaction between factors, as seen in Table 14. Thus, it can be stated that such factors have statistical significance regarding the time for performing change metric.

Concerning the impact of change metric and applying the F-Test with 0.05 of significance level, the null hypothesis is rejected for all factors, as presented in Table 15. Thus, all factors have statistical significance regarding this metric.

At last, performing an F-Test with 0.05 of significance level in the correctness of change metric, the null hypothesis is only rejected for ontology factor (see Table 16). Hence, even though much of the allocation of variation is due to random errors, it can be stated that the ontology factor has statistical significance for the correctness metric.

5.4.3. Hypotheses verification
As explained previously, the ontology factor is statistically significant for all dependent variables. However, even though that, the research hypotheses raised in Section 3 still remain
unanswered. Following the experiment execution, the intention is to discover if there are differences between the ontology alternatives regarding the dependent variables; moreover, if there are differences, we also intend to discover which one of the alternatives is “better”.

To verify the research hypotheses, the confidence intervals (considering the 0.05 level of significance) of the ontology alternatives are firstly analyzed and then statistical tests are applied (if it is not possible to answer only by analyzing the confidence intervals) for each one of the dependent variables. The data of all three metrics (at least for one of the ontologies) are not normal (the Shapiro–Wilk test was applied), hence we decided to apply a non-parametric test to compare the ontology alternatives, i.e., the Wilcoxon Test.

Fig. 9 presents the confidence intervals comparing the time for changing Ontologies 1 (Wang et al.) and 2 (Tenório et al). As seen in this figure, the intervals of the ontologies are overlapped, thus it can not be stated (only by analyzing the confidence intervals) which one of the ontologies requires a higher time for changing.

Aiming to obtain a stronger statistical comparison on the time for performing change, we applied the Wilcoxon non-parametric test to it. The application of the test, considering paired samples and a 0.05 level of significance, returned a $p$-value of 0.0006058. As a result, the null hypothesis can be rejected ($p$-value < significance level) and we can state that the time for performing change in Ontology 1 (Wang et al.) being higher than in Ontology 2 (Tenório et al).

Fig. 10 presents the confidence intervals comparing the impact of changes in Ontologies 1 (Wang et al.) and 2 (Tenório et al). As seen in the figure, the intervals of the ontologies are overlapped, thus it can not be stated (only by analyzing the confidence intervals) which one of the ontologies requires a higher time for changing.

Aiming to obtain a stronger statistical comparison on the time for performing change, we applied the Wilcoxon non-parametric test to it. The application of the test, considering paired samples and a 0.05 level of significance, returned a $p$-value of 0.001766. As a result, the null hypothesis can be rejected ($p$-value < significance level) and we can state that the impact of change in Ontology 1 (Wang et al.) being higher than in Ontology 2 (Tenório et al).

Table 14
Signification of the effects on the time for changing variable.

<table>
<thead>
<tr>
<th>$\alpha$</th>
<th>$\beta$</th>
<th>$z\beta$</th>
<th>$\epsilon$</th>
</tr>
</thead>
<tbody>
<tr>
<td>SS</td>
<td>7,206,297,380,726.72</td>
<td>146,087,975,439.03</td>
<td>675,670,820,113.32</td>
</tr>
<tr>
<td>MS</td>
<td>7,206,297,380,726.72</td>
<td>11,237,536,572.23</td>
<td>51,974,678,470.26</td>
</tr>
<tr>
<td>F</td>
<td>514.62</td>
<td>0.80</td>
<td>3.71</td>
</tr>
<tr>
<td>F-Table</td>
<td>3.88</td>
<td>1.76</td>
<td>1.76</td>
</tr>
</tbody>
</table>

Table 15
Signification of the effects on the impact of change variable.

<table>
<thead>
<tr>
<th>$\alpha$</th>
<th>$\beta$</th>
<th>$z\beta$</th>
<th>$\epsilon$</th>
</tr>
</thead>
<tbody>
<tr>
<td>SS</td>
<td>2170.85</td>
<td>48234.38</td>
<td>2221.28</td>
</tr>
<tr>
<td>MS</td>
<td>2170.85</td>
<td>3710.34</td>
<td>170.87</td>
</tr>
<tr>
<td>F</td>
<td>863.27</td>
<td>1475.47</td>
<td>67.95</td>
</tr>
<tr>
<td>F-Table</td>
<td>3.88</td>
<td>1.76</td>
<td>1.76</td>
</tr>
</tbody>
</table>

Table 16
Signification of the effects on the correctness of change variable.

<table>
<thead>
<tr>
<th>$\alpha$</th>
<th>$\beta$</th>
<th>$z\beta$</th>
<th>$\epsilon$</th>
</tr>
</thead>
<tbody>
<tr>
<td>SS</td>
<td>1.00</td>
<td>0.002</td>
<td>0.39</td>
</tr>
<tr>
<td>MS</td>
<td>1.00</td>
<td>0.0002</td>
<td>0.03</td>
</tr>
<tr>
<td>F</td>
<td>8.71</td>
<td>0.001</td>
<td>0.26</td>
</tr>
<tr>
<td>F-Table</td>
<td>3.88</td>
<td>1.76</td>
<td>1.76</td>
</tr>
</tbody>
</table>

In Fig. 9 the confidence intervals comparing the correctness of change in both ontologies are presented. As seen in the figure, the intervals of the ontologies are overlapped. Thus, it can not be stated which one of the ontologies presents a better correctness only by analyzing the confidence intervals. In this sense, we applied the Wilcoxon Test considering paired samples and a 0.05 level of significance, which returned a $p$-value of 0.73666. As a result, the null hypothesis can not be rejected ($p$-value > significance level) and we can state that the correctness of both ontologies are equal. One can note that this result was expected, since both confidence intervals include 0.

6. Discussion

The empirical study, proposed and explained in this paper, provides a set of useful conclusions regarding the comparison of two ontologies for representing features models in changing scenarios – one of them is based on OWL classes and properties and the other one is based on OWL individuals. In fact, the results obtained with the execution of this experiment allowed us to state: (i) there is significance on the effects of the ontology factor in all three dependent variables, (ii) the time for performing change on the ontology of Wang et al. is higher than on the ontology of Tenório et al., (iii) the structural impact of changes on the ontology of Wang et al. is higher than on the ontology of Tenório et al., and (iv) there is no statistical difference between the ontologies regarding the correctness of changes.

It is likely that the time to make changes on the ontology of Wang et al. is greater than on the ontology of Tenório et al. due...
Generally speaking, performing changes in the ontology of Wang et al. requires a greater number of changes. Moreover, the results of the experiment regarding the impact of change metric were somewhat expected, since changes in the ontology of Tenório et al. do not demand changes in the structure of the ontology (which happens in all changes made to the ontology of Wang et al.). Nevertheless, we keep this metric to obtain results from experimentation to confirm our previous expectation that it occurs.

Although the dependent variables analyzed (mainly the impact of change or flexibility) are important in evaluating which ontology-based feature modeling style seems to be more suitable in the context of changes at runtime, this experiment considers change scenarios in development time. Anyway, we argue that if an ontology is considered flexible, i.e., it demands little or no impact of change in development time on the ontology structure (e.g., the ontology of Tenório et al.), it is already an indication that such ontology is more suitable to be used in dynamic software product lines.

The results obtained from the experiment provide an important guide for the use of ontology-based feature modeling in traditional SPLs that constantly demand change/evolution requests in their features. In fact, to obtain more conclusions about the use of ontology-based feature models in the context of DSPL, it is necessary to design and execute new experiments considering several scenarios of changes at runtime. Moreover, these new experiments should also compare DL-based approaches with others strategies for performing automatic analysis of features models (e.g., propositional logic based approaches and constraint programming approaches).

In the following sections, the threats to the validity of the experiment and a comparison of this work with other ontology-based feature model approaches are discussed.

### 6.1. Threats to validity

This section describes concerns that must be improved in future replications of this study and other aspects that must be taken into account in order to generalize the results of the experiment performed in this work. In general, the design of the experiment aimed at minimizing a lot of the threats discussed in this section by randomizing the order of treatments application. However, there are threats that should be considered. To organize this section, the threats to validity were classified using the Internal, External, Construct and Conclusion categories (Wohlin et al., 2000). However, no threats considering the Construct validity could be identified.

#### 6.1.1. Internal

As the experiment involves the active participation of humans, it was prone to a number of internal threats, such as (i) history – it is possible that the moment at which the experiment occurred may have affected the results, however, this threat was minimized by previously scheduling the date of the experiment with all participants; and (ii) maturation – since the participants took around two hours to finish all the tasks of the experiment, it is possible that they were bored or tired during the last tasks.

#### 6.1.2. External

The sample of the experiment is only representative to the academic context of a single university in Brazil, thus there is an interaction of setting and treatment threat. In fact, it is difficult to generalize the results of the experiment to other contexts. The setting of the experiment must be broadened to obtain more generic results.

#### 6.1.3. Conclusion

During this study, we chose to work with a feature model of a non-real SPL, the Smartphone. Some might see the use of a non-real SPL as a potential threat to our results. Furthermore, due to cost restrictions – the experiment involves the active participation of people – the sample size of the experiment was 10 participants (repetitions), thus, there might be insufficient statistical power on the effects of the experiment. Finally, it is possible that random irrelevancies have occurred in the experimental setting, e.g., noise, distractions and so on.

![Confidence interval of the impact of change](image1)

![Confidence interval of the correctness of change](image2)
6.2. Related work

Many works have been used description logic based approaches for representing and analyzing feature models. They are mainly motivated by the reasoning efficiency capabilities (always decidable) provided by the use of DL reasoners. Moreover, the current literature still lacks an in-depth investigation of description logic-based approaches regarding analysis operations in feature models (Benavides et al., 2010; Benavides et al., 2013). Thus, aiming to contribute to this challenge, we conduct a controlled experiment which intends to identify an ontology-based feature modeling style best suited for software product lines that require some kind of flexibility, for instance DSPL. We take into account changing scenarios in feature models as analysis operations to measure flexibility, impact of changes and correctness of changes.

Thus, we divide the related research studies of our work in two groups: first, studies related to using DL based approaches to specify feature models (either by using class-based or by using individual-based modeling styles), and second, those related to using ontology-based feature modeling in approaches for dynamic reconfiguring feature models.

Regarding the first group, as already explained, Wang et al. (2007) present a technique to design ontology-based feature models. In that work, the feature model is represented by OWL classes and properties and OWL reasoning engines are used to check for inconsistencies of feature configurations automatically. However, their ontology was not evaluated in scenarios of changes, thus they do not present any empirical evidence about the evolution capabilities of this ontology.

Tenório et al. (2014), Filho, Barais, Baudry, Viana, and Andrade (2012) and Zaid et al. (2009) present ontologies for feature modeling using OWL individuals. The first work proposes a set of SPARQL queries that can be executed to reconfigure SPL products specified in their OntoSPL ontology. The second work proposes an approach to automatically verify consistency of ontology-based feature models using OWL individuals. The later presents an approach for semantic enrichment of SPLs using ontologies aiming to provide informal retrieval, inference and traceability properties to SPL development lifecycle. However, none of them present empirical results about the suitability (e.g., flexibility metric) of the ontologies in changing scenarios.

Lee, Kim, Song, and Baik (2007) use ontologies to represent feature models to analyze their variability and commonality. They use ontologies to analyze the semantic similarity of the feature model and use a class-based modeling style. Noorian, Ensan, Bagheri, Boley, and Biletskiy (2011) use description logic to identify inconsistencies in feature models and identify inconsistencies in products configured from the product line as well as to propose possible corrections. They implement their approach in a framework that uses OWL–DL (class-based modeling style) to represent feature and their configurations, and Pellet (Sirin et al., 2007) as reasoner. However, as we focus on the empirical comparison of ontology-based modeling styles in changing scenarios in order to guide the selection of a suitable ontology for DSPLs, both works do not present any empirical evidences on how their ontologies behave in changing scenarios.

Furthermore, Guo, Wang, Trinidad, and Benavides (2012) present an approach to deal with inconsistencies in FM evolution scenarios. They formalize such models from an ontological perspective and define constraints that must be satisfied in FMs to be consistent. Asadi, Gasevic, Wand, and Hatala (2012) investigate the use of ontological theories (i.e., Bunge’s ontology) for theoretical analysis of variability languages, for instance, feature models and Orthogonal Variability Models (OVM) (Pohl et al., 2005). Although both works rely on ontological concepts to deal with feature model, they neither provide OWL implementations for representing feature models nor choose a modeling style. Hence, they do not present any empirical result on how their ontologies can be used in changing scenarios.

Rincón, Giraldo, Mazo, and Salinesi (2014) propose an ontological rule-based approach to analyze dead and false optional features in feature models. They construct a feature model ontology in the individual-based style modeling and formalize rules for identifying dead an optional features in such ontology. However, as well as other related works for the first group, they do not present empirical results about the suitability of the ontology for supporting changing scenarios.

For the second group, Kaviani et al. (2008) enrich feature models using ontologies by annotating some features of a feature model with information (e.g., Non functional properties) contained in an existing ontology in the context of ubiquitous environments. 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. This work relies on the ontology-based feature modeling proposed by Wang et al. (2007). In the same direction, Bošković et al. (2010) complement the ontology-based feature modeling approach of Wang et al. (2007) with automated staged configuration. In this approach, a product configuration is performed as a set of consecutive steps and they provide an algorithm for automatic feature model specialization based on description logic reasoning mechanisms. Although the contributions of these studies to support reconfiguration of feature models, they do not present any empirical evidences regarding the flexibility of the ontologies used in the studies.

7. Concluding remarks

In this paper, we empirically compared two alternative approaches on ontology-based feature modeling. Such approaches use different modeling styles, one based on OWL classes and properties (Wang et al., 2007) and the other based on OWL individuals (Tenório et al., 2014). We conducted a controlled experiment in changing scenarios to verify if there were statistical differences regarding three metrics (i.e., time for performing changes, structural impact of changes and correctness of changes) in feature models specified in both approaches.

The main contribution of this work is to investigate, through the use of a rigorous controlled experiment, the capabilities to handle changes in feature models of approaches that follow two distinct ontology-based modeling styles to represent such models. This empirical study advances the current state of the art on feature model knowledge management by providing evidences that can be valuable to select a proper way (by using the power of DL based reasoners) to automatic analyze feature models using ontologies and to reconfigure products in domains that constantly need to change/evolve (e.g., DSPL). Moreover, as we depict in detail how this experiment was conducted, it can be extended without much design effort in order to compare other ontology-based feature modeling approaches.

Our results indicate that, with a 95% confidence level, using OWL individuals is more flexible and demands less time for changing than the one based on OWL classes and properties. We also found from the collected data that there is no statistical difference between the ontologies compared regarding the correctness of changes metric.

A possible limitation of our work may be considering only one approach on each ontology-based feature modeling style in the controlled experiment. However, we believe that the selected approaches are representative to the ontology-based feature modeling approaches, since some related works that follow the class-
based modeling style use the ontology proposed by Wang et al. (2007). In addition, some related works that follow the individual-based style are very similar to the ontology proposed by Tenório et al. (2014).

This work can be considered a first step towards selecting a suitable way for formalizing feature models knowledge under the perspective of ontology-based feature modeling to be used in the context of DSPLs. Such knowledge can be used to represent the concepts of DSPL domain as well as its relationships in a way that allows automated reasoning on feature models. Hence, it could also support reconfiguration of products.

We still need more evidences to better understand how ontology-based approaches can aid feature model evolution, as required by DSPLs. Our future works include conducting additional experiments to investigate how flexible other ontology-based feature modeling approaches are in changing scenarios in comparison to the approaches considered in this work. We also plan to design a new controlled experiment to incorporate dynamic reconfiguration scenarios that could be specified, for instance, in SPARQL queries. In this new empirical study, we intend to measure feature models capabilities on automatic consistency checking and reasoning efficiency considering the two distinct ontology modeling styles. We also expect to use the results of the new experiment, in addition to the evidences obtained in this work, to select an adequate ontology to represent feature models for DSPL purposes. Finally, we plan to develop a comprehensive knowledge-based software component based on such ontology to be incorporated into DSPL architectures in order to support automatic management of feature models and reconfiguration of products at runtime.

Acknowledgements

This work has been supported by the Brazilian institutions: Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) and Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES).

References


