A Verification Mechanism of Feature Models for Mobile and Context-Aware Software Product Lines

Fabiana G. Marinho* and Rossana M. C. Andrade
Graduate Program in Computer Science
Federal University of Ceará
Fortaleza, Brazil
{fabiana, rossana}@great.ufc.br

Cláudia Werner
Systems Engineering and Computer Science Program
Federal University of Rio de Janeiro
Rio de Janeiro, Brazil
werner@cos.ufrj.br

Abstract - Software Product Lines (SPLs) have been used to develop mobile and context-aware applications, which provide services and data for their users from anywhere and at any time using context information. In SPLs, commonality and variability of a system family are identified and often documented in a feature model. However, the development of a feature model for mobile and context-aware SPLs is not trivial, since it should comprise system and context information. Furthermore, the consistency check of feature models in the considered domain is also complex and demands advanced skills of software engineers. This paper proposes a mechanism to formalize and verify the correctness and consistency of feature models for mobile and context-aware SPLs based on a profile enriched with OCL specifications.

Keywords - Mobile and Context-aware domain, feature model, software product lines, consistency checking, OCL.

I. INTRODUCTION

The technological advances in mobile devices are fostering the creation of highly distributed and interactive applications, characterized by the dynamicity and uncertainty of resources. Requirements such as mobility and context-awareness demand interoperable, uncoupled, adaptable, and autonomous programming abstractions [1]. In these applications, human activities are extended by means of computational services that can adapt their behavior to the context in which they are inserted in a transparent way to the users [2].

The environment, user requirements and interfaces between software and hardware may change dynamically during the execution of this kind of application, requiring a response to these changes [3]. To enable adaptation without user intervention, the software, called mobile and context-aware in this paper, must be able to use the environment (physical or computational) and user information to provide relevant services or information [4].

Considering the aforementioned scenario, the development of mobile and context-aware software has become increasingly complex. On the other hand, Software Product Line (SPL) is a reuse-driven development paradigm, which has been successfully applied in various domains.

SPLs propose systematic software development based on a family of products that share commonalities and variabilities [5]. In [3], Hallsteinsen et al. affirm that SPLs have shown to be an efficient way to handle the requirements from the mobile and context-awareness domain.

In SPLs, a feature model is often used to capture and manage commonalities and variabilities, but there are many variant notations suggested by different authors, such as in [6], [7], [8], [9], [10], and [11]. As a matter of fact, variability is a key concept for SPLs, since it increases their flexibility and capacity to adapt according to user needs. In this sense, it is important to formalize and represent concepts related to variability in a consistent way to minimize variability modeling problems, especially in the mobile and context-awareness domain, where the construction of feature models is not trivial. In the considered domain, the feature model should capture the context information that may impact the variability modeling, according to changes in the execution environment. Furthermore, it involves complex requirements such as mobility, context-awareness, and adaptability.

Moreover, inconsistencies can arise between system features and feature adaptations triggered by changes in context. Whenever these inconsistent combinations are not considered, unexpected system behavior may emerge. Therefore, it is important not only to build a mobile and context-aware SPL, but also to create a verification mechanism to check the correctness and consistency of these SPLs and their configured products.

There are many consistency checking approaches for feature models in literature [12]. For example, Zia di et al., [8] present a UML profile to model variability and suggest the use of OCL [13] to ensure some structural constraints. However, this approach is not suitable for the considered domain and it is necessary to invest efforts for modeling mobile and context-aware SPL. An attempt in this direction is done by Fernandes et al. in [14] with the UbiFEX-Notation. Our proposal extends these works, including concepts of domain driven design and formal methods to minimize errors and inconsistencies in feature models. We focus on feature models because they are used since early stages of SPL engineering, so any error in this model will inevitably affect the configured models and later the final products.

* Doctorate student in Computer Science - CNPq process number 552924/2008.3
To explore the viability of this assumption, we propose a verification mechanism to check, at development time, feature models for the mobile and context-awareness domain based on a UML 2.3 profile [15][16] enriched with OCL specifications. UML was chosen to model variability as it is considered a “de facto” standard, both in industry and academy. Besides, UML provides extension mechanisms that allow its customization to specific domains. Hence, it is possible to develop a specific modeling language to build feature models for mobile and context-aware SPLs.

In short, the major contribution of this paper is to increase the quality of feature models by means of the proposed verification mechanism, and consequently improves the quality of mobile and context-aware SPLs. This verification mechanism checks, at development time, the correctness of feature models in the considered domain and the consistency of their configurations. Furthermore, a tool will be developed to support the proposed verification mechanism, given that conventional tools (e.g., Magic Draw [17], Papyrus [18]) that support profiles and OCL specifications, do not provide support for some specific tasks for mobile and context-aware SPLs, such as product configuration and context modeling.

The remainder of this paper is divided as follows. In Section 2, we discuss related work to the proposed mechanism. In Section 3, we describe some concepts related to UML extension mechanisms and present the steps to develop a profile to build feature models for mobile and context-aware SPLs. In Section 4, we describe the approach adopted in the verification mechanism for feature models. In Section 5, we apply the proposed verification mechanism to analyze its usefulness, and, in Section 6, we present our conclusions and future directions.

II. RELATED WORK

According to Benavides et al. [12], some existing approaches check the consistency of feature models based on rigorous mathematical theories. For example, [19], [20], and [21] propose feature model translation into propositional formulas. Czarnecki and Wa [22] propose the extraction of feature models from propositional formulas. The use of constraint programming is investigated in [23] and [24]. Their proposals do not focus on mobile and context-awareness domain, then, they do not tackle the context information and are not suitable for this domain.

Other studies use less rigorous formal elements that facilitate the use of non-specialist users. Zaid et al. [25] and Wang et al. [26] propose ontology to formalize feature models to check model consistency and conflict detection through pre-defined rules. Ziad et al. [8] propose an approach based on a UML profile for SPLs and the use of OCL to ensure structural constraints of feature models. These authors do not consider context information like [25] and [26]. However, since [8] also focus on UML profiles and OCL, we take their work into account and extend it to the mobile and context-awareness domain.

Some research works address context-awareness. For example, ContextUML [27] is one of the first Domain-specific Language for context-awareness (DSM). A DSM is a software engineering methodology for designing and developing systems based on a domain-specific language (DSL), which is a programming language or specification language dedicated to a particular problem domain, a particular problem representation technique, and/or a particular solution technique [28]. ContextUML is a DSM which presents a context-awareness metamodel for Web services. Cappucine [29] describes a Model-Driven Engineering (MDE) approach for producing SPLs according to context information. However, this approach is more related to adaptation mechanisms. These works tackles context-awareness with the MDE approach, and, therefore, are used for transformation purposes.

In this study, we focus on consistency checking of feature models for mobile and context-aware SPLs. Nowadays, there are few studies in literature that take context and consistency checking into account during variability modeling. For example, [30] propose the KobrA method, where context models are described using analysis of specific systems and the model built is specific for the analyzed system. In this approach, no consistency checking is proposed. Fernandes et al. [14] propose a notation for variability modeling for context-aware SPL, called UBIFEX-Notation. The authors also propose the UBIFEX-Simulation to minimize inconsistencies in the configured context-aware models. However, the correctness of the built models is not checked and no formalism is considered.

Ubayashi et al. [31] propose context-dependent SPL Engineering using a formal method. In this method, context is treated as a separate SPL, and context specifications are built as core assets. The authors use formal methods to specify and check the correctness of the assets constructed, but no verification is proposed for the configured products.

The approach proposed in this paper aims to provide solutions to the gaps found in these works. In Table I, we show a comparative analysis of our proposal and the works discussed.

<table>
<thead>
<tr>
<th>Approach</th>
<th>Context Modeling</th>
<th>Model Verification</th>
<th>Configuration Verification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mannon [19]</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Zhang et al. [20]</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Batory et al. [21]</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Czarnecki and Wa [22]</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Benavides et al. [23]</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>Trinidad et al. [24]</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Zaid et al. [25]</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Wang et al. [26]</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Ziad et al. [8]</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Atkinson et al. [30]</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Fernandes et al. [14]</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Ubayashi et al. [31]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Our proposal</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

Our proposal allows an explicit representation of the context information in the feature model. Furthermore, the
proposed verification mechanism is intended to ensure, at development time, the correctness of feature models, and anticipate problems that may occur in the configurations of this feature model at runtime. For this, a UML 2.3 profile with constraints specified in OCL is used to formalize well-formedness rules for these models.

III. ACTIVITIES FOR THE UML PROFILE ELABORATION

The principle of extensibility allows the use of UML for modeling systems that have specific needs. UML can be extended in two ways: (i) the lightweight form; and (ii) the heavyweight form. In the heavyweight form, a new modeling language is proposed by extending MOF (Meta-Object Facility) [15]. The lightweight form implies extending UML through stereotypes, tagged values and constraints, which are grouped in a profile to specify peculiarities of the modeled domain [15][16].

We adopt the lightweight extension mechanism in this paper. Therefore, we propose a UML profile, which is developed according to the guidelines for the specification of UML profiles suggested in [32]. The extensions defined in this profile are based on UML 2.3, as mentioned in Section II, and they only concern UML class diagrams.

In this paper, a feature model for mobile and context-aware SPLs consists of two models: a System Model and a Context Model. A System Model captures commonalities and variabilities of a family of products. A Context Model captures contextual information that is relevant to the modeled domain.

Although there are several notations available for system models and context models, we decided to propose a new one, based on [8][14], which can be manipulated by our verification mechanism without restrictions. In general, the tools available do not provide full access to their resources, which hinders their extension.

Four activities were performed in this work to build the UML profile for System Models and Context Models in the considered domain: Requirements Elicitation, Metamodel Elaboration, UML Profile Development, and Stereotype Definition.

A. Requirements Elicitation

Kang et al. [35] proposed the first model to capture SPLs variability; however, after years of research, there is still not a consensus, and several extensions to this model were proposed. Considering this scenario, [39] identified a set of requirements for building a System Model for SPLs from a compilation of several works in literature. This work was later refined by [42].

On the other hand, a large number of context-aware applications have been developed over the years for a variety of application domains, using different context models, for example [43] and [44]. These experiences have influenced the establishment of a set of requirements that must be observed during the context modeling.

In this paper, we revised the set of requirements proposed by [45] to identify requirements that must be satisfied by a notation to express Context Models for mobile and context-aware SPLs, as follows.

- **Context elements**: context elements can influence product configuration in an SPL. A graphical representation must clearly express context elements.
- **Context attributes**: context attributes store context information to characterize context elements. A graphical representation must clearly express context attributes.
- **Context triggers**: context triggers represent how a context situation previously defined can influence product configuration in an SPL. It is desirable that a graphical representation express context triggers.
- **Information source**: context information can be provided from heterogeneous sources, such as user provided, profile provided, sensor provided, and derived from other context information. A graphical representation must clearly express the source of context information.
- **Information validity**: the validity of context information can be permanent, rare, frequent, and volatile. A graphical representation must clearly express the validity of context information.
- **Information quality**: context information can have variable levels of quality due to its dynamism and heterogeneity. A graphical representation must clearly express the quality of context information.
- **Relationships between context information**: there are several relationships between the types of context information. It is desirable that a graphical representation expresses these relationships.
- **History**: applications may need to access past states of context information. The management of historical context can become complex if the number of updates is high. Thus, it may not be feasible to store all values in a graphical representation. And,
- **Rationale**: context-aware applications usually require a rationale to evaluate if any context change occurred to decide if an adaptation is necessary. It is desirable that the graphical representation supports context reasoning techniques.

B. Metamodel Elaboration

The metamodels express the elements of a domain, the relationships between these elements and the constraints that govern their structure and behavior. The feature model is not part of the set of models defined by UML, so it does not have a metamodel specification defined by OMG. Therefore, we used the previously elicited requirements to formalize two metamodels: a metamodel for System Models and a metamodel for Context Models.

Fig. 1 depicts meta-classes and relationships used to capture a System Model. An important requirement for a notation that represents System Models is the representation of
dependence and mutual exclusivity between model elements. This requirement is achieved using composition rules. A composition rule has the structure: antecedent operator consequent. The operator can assume the value requires or excludes.

The antecedent and the consequent are expressions, which can be literals or booleans, and denote a feature of the domain or a combination of features. The composition rule that expresses dependence between features is called Inclusive Rule, while the composition rule that expresses mutual exclusivity is called Exclusive Rule [42].

The meta-class Composition Rule captures the Inclusive Rule and the Exclusive Rule. The meta-class Composition Rule has two composition relationships that represent the antecedent and the consequent parts of a rule. The meta-class Expression is associated to the meta-class Element, implying that composition rules are composed of expressions, which are composed of elements.

Fig. 2 depicts meta-classes and relationships used to capture a Context Model. The meta-class Context Element represents the relevant context elements that can influence the product configuration of an SPL. The Context Attribute represents the attributes of a Context Element. The meta-class Context Attribute has five attributes: Name, Value, Validity Kind, Source Type, and Quality.

The attribute Value is used to store the attribute value and can assume the types Integer, String, Float or Boolean. The attribute Validity Kind represents the update frequency of context information (permanent, volatile, frequent, and rare). The attribute Source Type represents how the context information is obtained (informed by the user, derived from other information, obtained from a profile or obtained from a sensor). The attribute Quality represents the context information reliability (high or low).

The meta-class Trigger Rule captures the context situations that may cause a configuration in the System Model.

This meta-class is associated to one or more meta-class Expression, which represents an E-C-A (Event-Condition-Action) Rule. The Event corresponds to a value or a combination of values of Context Attribute. The Expression of an Event that can be obtained from the relational operators (>, <, >=, <=, =, < >) and logical operators (OR / AND).

The Condition states that if the Expression that defines the Event is evaluated as True, then the Expression that composes the Action must be evaluated. The Action consists of a feature or a combination of features belonging to the System Model. Logical operators (AND or NOT) can be used in the Expression that composes the Action. The presence of a feature in the Expression of the Action indicates its addition in the System Model and the presence of the NOT logical operator indicates the removal of the feature from the System Model.

C. UML Profile Development

In this paper, we propose two profiles to build correct feature models for mobile and context-aware SPLs. UML meta-classes are identified through the stereotype `metaclass` and profile specific stereotypes are identified with the stereotype `stereotype`. The first profile includes specific stereotypes and constraints to build System Models (Fig. 3).

The stereotype `Optional` was defined to specify optional elements and the stereotype `Mandatory` to specify mandatory elements. A `Variation` is composed by one or more `Variant` and a `Variant` is associated to one and only one `Variation`. The stereotype `Variant` also extends the stereotype `Optional`, since a `Variant` can be selected or not in a product configuration. A `Variation` can be exclusive (XOR) or optional (OR). These constraints and the composition rules defined to the System Model are domain-specific and we used OCL constraints to specify them in the UML profile.

We used the Mobile and Context-Aware Visit Guide SPL to illustrate the construction of a System Model. This SPL produces mobile and context-aware visit guides that run on the visitor's mobile device and provide information about the
environment, people around, and environments that are visited. The behavior of these functionalities can be adapted according to the visitor’s current context, which includes the visitor’s location, his/her profile/preferences, the characteristics of his/her mobile device, and information concerning other people present in the same room.

To define the SPL scope, we identified a collection of applications that fit into this domain and organized relevant information using a System Model. The main characteristics identified were Dynamic Execution Environment, Adaptability, and Context-Awareness. We also distinguished common features derived from the distributed nature of these applications, such as Message Exchange, Service Description, and Service Discovery [1]. Then, we created the System Model using the proposed UML profile. Due to space restrictions, Fig. 4 only shows part of this System Model.

The Service Description feature is formed by the Syntactic Description feature, which is implemented by the Keyword Service Description. Additionally, the Service Discovery feature is implemented by the Application Layer and Centralized Service Discovery. The Message Exchange has a variation in the Exchange Type, which can be implemented as Synchronous using Remote Procedure Call, or implemented as Asynchronous using Tuple Space or Event Based. Finally, the Context Management feature has three mandatory features, namely Capture, Acquisition, and Inference, as well as an optional feature called Persistence.

We specified a composition rule using OCL as an example to this Mobile and Context-Aware SPL (Table II).

**TABLE II. EXAMPLE OF A COMPOSITION RULE OF THE SYSTEM MODEL**

<table>
<thead>
<tr>
<th>Inclusive Rule: Service Discovery Centralized requires Service Description by Keyword</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base Class: Model</td>
</tr>
<tr>
<td>Constraint:</td>
</tr>
<tr>
<td>inv:</td>
</tr>
<tr>
<td>self.ownedMember-&gt;exists(el:NamedElement</td>
</tr>
<tr>
<td>implies self.ownedMember-&gt;exists(el:NamedElement</td>
</tr>
</tbody>
</table>

The abstract stereotype «Context Association» is used to classify associations between the stereotypes «Context Element» and «Context Attribute». This is necessary to capture the requirements defined for context models. The stereotype «Source» is represented by the stereotypes «Sensed», «User», «Derived», and «Profile». The stereotype «Sensed» denotes context information obtained from sensors. The stereotype «User» indicates that the user provided the context information. The stereotype «Derived» is used in associations between context elements derived from other context elements. The stereotype «Profile» represents context information obtained from a user profile.

The stereotype «Validity» is represented by the stereotypes «Permanent», «Rare», «Frequent», and «Volatile». The stereotype «Permanent» symbolizes context information that does not change with time. The stereotype «Rare» symbolizes context information that undergoes change with very low frequency. The stereotype «Frequent» symbolizes context information that changes value frequently, and the stereotype «Volatile» symbolizes context information that changes with a high frequency. Stereotype «Quality» is represented by the stereotypes «High» and «Low». The stereotype «High» symbolizes reliable context information and the stereotype «Low» symbolizes context information with low reliability.
The domain of Mobile and Context-Aware Visit Guides is highly dependent on context modeling. Fig. 6 depicts a piece of the context model developed to capture context information necessary to the Mobile and Context-Aware Visit Guide SPL.

**TABLE III. EXAMPLE OF A TRIGGER RULE OF THE CONTEXT MODEL**

<table>
<thead>
<tr>
<th>Trigger Rule:</th>
<th>if the Network Type assumes Cellular Network, it implies that the Service Discovery must use the Centralized feature.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Base Class:</strong></td>
<td>Model</td>
</tr>
<tr>
<td><strong>Constraint:</strong></td>
<td>if umlExamplePackage::Context Model::Network.type.value='Cellular' then self.ownedMember-&gt;exists(el:NamedElement</td>
</tr>
</tbody>
</table>

### D. Stereotype Definitions

The stereotypes included in the UML profiles to model variability for mobile and context-aware SPLs were defined based on the structure suggested in [16]. Therefore, for each stereotype, we define a name, base class (from UML metamodel), and a constraint.

The association of OCL constraints to stereotypes of a profile reduces the likelihood of modeling errors, as these constraints are checked whenever the stereotype is applied to a model. Due to space limitations, only some examples of the stereotypes defined in the proposed UML profile are described in Tables IV and V. We selected one example from the System Model profile and one example from the Context Model profile to show in this paper.

### IV. THE VERIFICATION MECHANISM

According to Moura [40], the verification of feature models means the formal process of determining whether or not a model satisfies a set of well-defined criteria. Literature review shows that several methods and techniques have already been proposed, especially regarding feature oriented modeling notations [12][41].
It is important to mention that feature model correctness includes both structural and semantic correctness of the models. On the other hand, feature models consistency is related to uniformity and lack of contradictions between feature models and their configurations [40].

The proposed verification mechanism of feature models for Mobile and Context-Aware SPLs consists of three main steps: Model Verification, Rule Verification and Configuration Verification. This mechanism aims to verify at development time:

- the correctness of System Models and Context Models;
- the correctness of composition rules defined for the System Models and trigger rules defined for the Context Models;
- the consistency between composition rules defined for the System Models and between trigger rules defined for the Context Models; and
- the consistency of the configured System Models.

**A. Model Verification**

In the Model Verification we check the correctness of the System Model and the Context Model. However, to enable the correctness checking of these models, well-formedness rules should be established to act as guidelines for the verification mechanism. In the proposed verification mechanism, the well-formedness rules are defined by OCL specifications associated to UML stereotypes in the constructed profiles. Then, the verification mechanism monitors the feature models construction and sends notifications every time the well-formedness rules of a model are infringed.

**B. Rule Verification**

The Rule Verification step covers the correctness checking of composition rules and trigger rules and the consistency checking between these rules. For the composition rules, checks to be performed are the following:

- If the feature is part of the antecedent of a composition rule it is necessary to verify if this feature belongs to the System Model;
- If the feature is part of the consequent of a composition rule it is necessary to verify if this feature belongs to the System Model;
- If the feature is part of the antecedent of an Inclusive Rule, it is necessary to verify if the feature belongs to the consequent is present in the System Model;
- If the feature is part of the antecedent of an Exclusive Rule, it is necessary to verify if the feature belongs to the consequent is not present in the System Model;
- If the feature is part of the consequent of an Exclusive Rule, it is necessary to verify if this same feature is not part of the consequent; and
- If the feature is part of the consequent of an Exclusive Rule, it is necessary to verify if the same feature is not part of the antecedent.

For the trigger rules, the checks to be performed are the following:

- if the feature is part of the antecedent of a trigger rule, it is necessary to verify if this feature belongs to the Context Model;
- if the feature is part of the consequent of a trigger rule, it is necessary to verify if this feature belongs to the System Model;
- if a trigger rule implies in the inclusion of a feature in the System Model, it is necessary to verify if there is another trigger rule that implies in the removal of the same feature from the System Model (thus, this verification should be performed if both rules are executed in the same scenario); and
- if a trigger rule implies in the removal of a feature in the System Model, it is necessary to verify if there is another trigger rule that implies in the inclusion of the same feature from the System Model (thus, this verification should be performed if both rules are executed in the same scenario).

**C. Configuration Verification**

In traditional approaches of SPLs, all variation points and features that are part of the configured feature model are defined at development time. In these cases, the consistency checking is performed as soon as the System Model configuration finishes.

However, for mobile and context-aware SPLs, some decisions are postponed until runtime, and according to changes in the context elements, new model configurations can occur. In this case, the proposed verification mechanism anticipates, at development time, possible inconsistencies that may occur at runtime, using a static technique to simulate scenarios that can result in feature model configurations.

In the static simulation technique used by the proposed verification mechanism, we manually perform a sequence of activities to check a given scenario. In the first activity, we assign values for the context attributes, according to the scenario we want to check the consistency. In the second activity, we examine the trigger rules specified in the Context Model based on the assigned values. The expressions that represent the Event of each trigger rule are evaluated to identify trigger rules for which the Event expression is evaluated as True for the specified scenario. For these trigger rules, we analyze the expressions that represent the Action and configure the System Model accordingly.

Next, the configured System Model should be checked to identify possible inconsistencies. A necessary verification consists of identifying inconsistencies related to features added.
For each feature added, the following checks are performed:

- if the added feature is part of the antecedent of an Inclusive Rule, it is necessary to check if the feature that belongs to the consequent is present in the System Model;
- if the added feature is part of the antecedent of an Exclusive Rule, it is necessary to check if the feature that belongs to the consequent is not present in the System Model;
- if the added feature is part of the consequent of an Exclusive Rule, it is necessary to check if the feature that belongs to the antecedent is present in the System Model and the feature that belongs to the consequent is not present in the System Model; and
- if the added feature is a variant, it is necessary to check if the maximum cardinality of the variation point is satisfied after the feature inclusion.

The verification mechanism also checks the following scenarios related to the features removed:

- if the removed feature is optional, no inconsistency is generated; otherwise, if the removed feature is mandatory, an inconsistent model is generated by configuration;
- if the removed feature is part of the consequent of an Inclusive Rule, the verification mechanism should identify the inconsistency, since the feature could not be removed from the model; and
- if the removed feature is a variant, it is necessary to verify if the minimum cardinality of the variation point is satisfied after the feature removal.

If a composition rule is not being respected, an inconsistency between the trigger rule and the composition rule is identified. Likewise, if the cardinality of a variation point is not being respected, an inconsistency between the trigger rule and the cardinality is identified.

It is important to remark that to guarantee a comprehensive consistency verification of configured feature models implies that the verification mechanism be executed for each scenario which may causes System Model configuration. Nevertheless, the proposed verification mechanism just identifies inconsistencies associated with scenarios pre-defined at development time, and it does not have the ability to create new scenarios at runtime. Therefore, to ensure the consistency checking of all possible scenarios is a complex task, since a scenario which was not foreseen at development time may occurs during runtime.

V. APPLICATION OF THE VERIFICATION MECHANISM

In this Section, we apply the proposed verification mechanism in the example presented in Section IV (Mobile and Context-Aware Visit Guide SPL) to analyze its usefulness in the construction of correct and consistent feature models for mobile and context-aware SPLs.

The Model Verification step aims to check feature model correctness using the UML profiles. The UML profile for System Models and the UML profile for Context Models checks if the constraints defined in the profile stereotypes are satisfied. Considering the System Model shown in Fig. 4, we verify that its structure is correct, since this model respect the restrictions associated to the corresponding profile. In the same way, the Context Model shown in Fig. 6 is correct.

An example of a situation that does not comply with the constraints defined in the System Model profile is when a mandatory feature is modeled as dependent of an optional feature. The configuration can produce an incomplete feature model, as there may be a situation where the mandatory feature is not selected for a given configuration, if the optional feature is not selected. In this case, the verification mechanism identifies that the well-formedness rule has not been observed in the model built.

In Rule Verification step, the composition rules and the trigger rules are evaluated to verify their correctness and consistency. These rules are specified using OCL constrains, but to simplify this example, we used their general structures defined in Section IV. Considering the System Model presented in Fig. 4, the Exclusive Rule ERI (Fig. 7) is incorrect and is making the System Model incorrect, since the feature Message Exchange will never be selected for any configuration, despite being classified as a mandatory feature.

An example of inconsistent trigger rules is shown in Fig. 8, where the trigger rule ECA2 cancels the action of the trigger rule ECA1.

At the Configuration Verification step, it is necessary to build an initial product configuration from the System Model of the Mobile and Context-Aware Visit Guide SPL (Fig. 9). To verify the correctness of this configured model, the System Model profile is applied to check if the well-formedness rules and composition rules are met. Analyzing the model obtained in Fig. 9, we realize that the well-formedness rules were not observed, since the mandatory feature Message Exchange was not selected. Besides, composition rules were not met. The configured model does not satisfy the Inclusive Rule presented in Table II, as the feature Centralized was selected, but the feature Keyword was left out. Therefore, the configured System Model is not correct.

To check the consistency of configured models caused by changes in context, it is necessary to use the proposed Static Simulation Technique. For the Context Element Mobile Device shown in Fig. 6, Table VI presents a combination of some inserted values for the Context Attributes Battery and
Memory. The associated values represent the percentage of the remaining battery and the percentage of memory used.

![Figure 9. Configuration of part of the System Model for Mobile and Context-Aware Visit Guide](image)

**TABLE VI.** VALUES FOR THE ATTRIBUTES BATTERY AND MEMORY.

<table>
<thead>
<tr>
<th>Battery</th>
<th>Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>0</td>
</tr>
<tr>
<td>80</td>
<td>50</td>
</tr>
<tr>
<td>50</td>
<td>80</td>
</tr>
<tr>
<td>30</td>
<td>100</td>
</tr>
</tbody>
</table>

Fig. 10 presents the trigger rule **ECA3** used in this simulation. In this trigger rule, the **Event** expression evaluates the context situations when the Mobile Device has low battery power and high memory consumption.

**EC3: (Mobile Device.Battery.value <= 70 AND Mobile Device.Memory.value >= 70) implies NOT (Centralized)**

![Figure 10. Example of a trigger rule used in the simulation technique.](image)

The **Event** expression is evaluated as True for the context attribute values shown in Table VII.

**TABLE VII. CONTEXT ATTRIBUTE VALUES FOR THE TRIGGER RULE ECA3.**

<table>
<thead>
<tr>
<th>Battery</th>
<th>Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>80</td>
</tr>
<tr>
<td>30</td>
<td>100</td>
</tr>
</tbody>
</table>

According to the values shown in the Table VII, the **Action** expression defined in **ECA3** states that the feature **Centralized** should be removed from the configured System Model. In this example, the simulation of a configuration caused by a context change (feature **Centralized** removal) did not cause any inconsistency, because the feature **Centralized** is optional.

Table VIII summarizes an analysis of the four objectives for the verification mechanism defined in Section V in relation to the steps performed using the example for the Mobile and Context-Aware Visit Guide SPL. According to Table VIII, all objectives of the verification mechanism were satisfied with the execution of the three steps. Therefore, the proposed verification mechanism contributes to increase the quality of feature models and consequently improves the quality of Mobile and Context-Aware SPLs.

**TABLE VIII. OBJECTIVES VERSUS VERIFICATION MECHANISM STEPS.**

<table>
<thead>
<tr>
<th>Objective</th>
<th>Model Verification</th>
<th>Rule Verification</th>
<th>Configuration Verification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model Correctness</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rule Correctness</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Rules Consistency</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Configured Models Consistency</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
</tbody>
</table>

**VI. CONCLUSIONS AND FUTURE WORK**

This work proposed a verification mechanism to check, at development time, correctness and consistency of feature models for mobile and context-aware SPLs. This verification mechanism comprises three main phases, as follows: the first is responsible for checking the correctness of System Models and Context Models; the second concerns the correctness and consistency verification of rules specified in these models; and the last activity corresponds to the correctness and consistency verification of the configured products of the SPL.

The proposed approach uses a set of UML extensions grouped in a UML profile to capture similarities, variabilities and context elements that influence products configured from an SPL. This profile was defined using UML 2.3 metamodel and was enriched with constraints specified using OCL to control the development of correct feature models.

As future work, we will conduct a case study to model and verify other feature models for mobile and context-aware SPLs in order to better analyze its benefits. Furthermore, we intend to support the proposed verification mechanism with a tool that will automate the construction and consistency checking of feature models based on the proposed UML profile.

**ACKNOWLEDGMENT**

Authors thank researchers and students involved in the MobiLine Project [46] for the fruitful discussions about mobile and context-aware applications and Software Product Lines.

**REFERENCES**


https://www.magicdraw.com/

http://www.papyrusuml.org


