Semantic Variability Modeling for Multi-staged Service Composition

Bardia Mohabbati[1,3], Nima Kaviani[2], Dragan Gašević[3]
[1]Simon Fraser University, [2]University of British Columbia, [3]Athabasca University, Canada
mohabbati@sfu.ca, nimak@ece.ubc.ca, dgaseavic@acm.org

Abstract

Feature models as the main modeling metaphors for software product line conceptualization are not expressive enough to cover all the variability needed to support adaptive engineering of service-oriented systems in highly dynamic environments. In particular, feature models lack required semantics to incorporate non-functional requirements (NFRs) and enable reasoning over the set of possible products in order to derive the best configurations. Ontology languages, as easily expandable semantically enriched conceptualization methodologies, can be used as the underlying languages for expressing feature models. This would allow augmentation of feature models with NFRs and would add the possibility for inference and reasoning to feature models. In this paper, we show how transformation of feature models to ontologies coupled with constraints over configuring products can help with reasoning over a product family and creating adaptive service compositions in an exemplified ubicomp application.

1. Introduction

The emergence of highly dynamic environments such as mobile systems and ubiquitous computing subsuming wide range of heterogeneous computing devices (e.g. handheld computers, PDAs, and smart phones), call for more flexible and adaptive development of software-intensive systems. Those systems now need to be composed, configured and delivered based on capabilities and resource constraints of the deployment platform of the target users. Accordingly, we require support to configure final software products, which provide the utmost functionality and satisfy non-functional requirements (NFRs) derived from the target deployment platforms.

Service-oriented Architectures (SOAs) come to the scene as a promising software architecture style for addressing the on-going challenges of the development of ubiquitous computing systems. In particular, platform independence, interoperability, loose-coupling, reusability, discoverability, composablity and dynamic binding are significant traits [2] for this context. However, to be able to software systems based on the SOA principles in the provided ubiquitous computing context, we need to equip developers with appropriate software methodologies, which can allow for effective software development process. This process involves the development throughout different abstraction layers comprising such as those already identified in the literature business process, service composition, service interface, and service implementation layers [1][3][4]. While each of the abovementioned layers calls for a lot of dedicated research, in this paper we focus on one key problem: How to develop service-oriented systems by considering the specificities of different target deployment platforms. That is, more concretely, how can we consider different device capabilities in the service-oriented development process? Given a proven track record of Software Product Line Engineering (SPLE) in the domain of mobile computing (e.g., Nokia as one of the most known examples), the idea of the use of software product line principles seems to be a first natural option for the problem under study. The SPLE discipline provides methods for managing variability and commonalities of core software assets in order to facilitate the development of families of software-intensive products. Here, software families are characterized by a set of features shared by each individual product of a family. At the same time, each specific product may have certain specificities coming from particular requirements of the product at hand. While in general we may have various different sources of variability, in this paper, we exclusively focus on the problem of variability coming from the different delivery platforms of service oriented systems. For example, two different types of mobile devices might support different connectivity or security protocols. This may directly impact a decision on which payment service to use in a concrete service-oriented system under development.

As it is reflected in the previously-mentioned four layers of abstraction reflects, the prevailing approach to developing service oriented architectures is based on business processes modeling. Once defined, business process models guide the process of service composition. This is the exact place where we take in to account of applying SPLE principles. In particular, we exploit feature modeling, which allows for identifying and managing the existing services and components in terms of common and variable features of a system in a product line. Furthermore, feature models provide structural management scheme of variability derived from NFRs of domain assets and to approach semi-automatic configuration and generation of products in response to the specific requirements for different products of the same service-oriented product family.

Aiming to provide a methodology, we propose a multi-staged specialization process of feature models [5]. In this process, we model service compositions by using feature modeling diagrams, which are then annotated with the NFRs, which each deployment platform needs to satisfy. In this process, we make use of the Ontology Web Language (OWL) and Semantic Web Rule Language (SWRL)[6][7]. These languages are used to represent feature models formally, so that we can automatically detect a set of allowed feature model specialization of a given target device. In addition, through using expandability and annotation capabilities of ontologies, NFRs of service in product lines can be formally incorporated into the ontological specification of feature models, expanded over time, or augmented with additional non-functional ontologies. In other words, the variability derivation points derived from NFRs of services is catered by the specification of the relevant declarative elements (device ontologies, service descriptions). Description Logic as the underlying logic for ontologies renders non-functional requirements to be formally introduced into the feature model ontologies. Furthermore, ontology-based semantics support logical reasoning over semantic descriptions, which in turn helps with validating (non-) functional requirements, intelligent product configuration, and product consistency check.

2. Motivation Example

Dealing with diverse services offered by various service vendors requires a proper mechanism to select appropriate services whose composition can lead to a desired functional system. The architectural structure for developing a system is thus influenced by possibilities in choosing the appropriate set of services from the list of existing ones. An adaptable design hence, should lay out all different possibilities for composing services in order to enable a system detect, find, and replace its set of services when needed. Consider an application developed to be launched on a large display at an airport to enable passengers look for information about flight schedules, stores at the airport, the city, etc. In an ideal ubiquitous environment, the application should enable passengers to connect to the application seamlessly and utilize its functionalities.

A common method of interaction would be through using cell phones carried by passengers. The application should enable every passerby to use his or her cell phone to connect to the application regardless of the type of phone s/he carries. Interaction and communication with the application on the large displays can happen through different communication protocols (e.g., WiFi, Bluetooth, IrDA), and different message exchange protocols (e.g., Http over TCP, Sockets, Channels, etc.). Consequently, the large display application needs to be able to adapt its communication and interaction protocols to the type of phone requesting a connection. Furthermore, the content delivered to the phone should also be adjusted to the physical specifications of the device (e.g., the display size, the resolution, etc.). The problem gets more complicated if part of the presentation or logic for the application needs to be delivered to the mobile phone. Adaptability becomes an important issue, since the application needs to incorporate different modules depending on the infrastructure for the mobile phone, while keeping the functionality consistent from phone to phone. Being able to properly lay out a design architecture that brings all these diversities in selecting components under a unified model, would facilitate selection of components through reasoning and would enable proper adaptation of the software with respect to the diversities imposed by joining and leaving mobile devices. We believe the support for specifying variabilities and commonalities provides proper modeling requirements to deal with the dynamicity required in ubiquitous environments.

3. Proposed Approach

As indicated in the introduction, the goal of this paper is to propose a methodology for developing families of service-oriented systems based on a multi-staged specialization of feature models. Figure 1 illustrates the overview of the overall development process. The first two stages adapt the existing feature-oriented domain analysis and design approaches (i.e., *Domain Analysis and Design*). Besides their adaptation of the service-oriented context, this stage also specifies NFRs related to the capabilities of the target deployment platforms. Once defined, the next step is to provide a specializa-

tion of the feature model first based on the target deployment platforms and second based on user preferences (i.e., *Service Adaptation* in Figure 1).

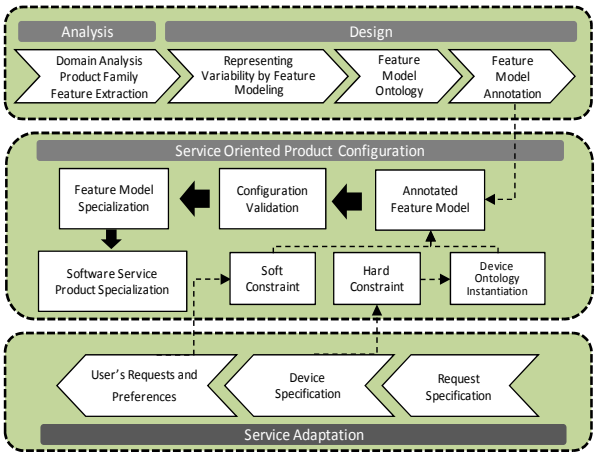

Figure 1. The overview of the steps of the multi-staged specialization of service compositions

In this paper, we only focus on the first step, where we are considering the characteristics of the target deployment platform. In the overall process, we make use of the semantic Web languages for rules and ontologies (OWL) and SWRL to automate the process of specialization process and obtain a set of feature model specializations for a given set of the requirements (i.e., *Service-oriented Product Configuration*). The rest of the section describes the proposed approach in detail.

3.1. Analysis and Design

The purpose of analysis and domain engineering is to develop domain assets, aka features, to identify and extract a set of reusable features involved in the process of service-oriented product family. There have been many methodologies established to perform domain analysis. These methodologies could be opted in the engineering of reusable architectures and components (e.g. Feature-Oriented Domain analysis (FODA) [10], Feature-Oriented Reuse Method (FORM)[11], FeatuRSEB [12]). An appropriate comprehensive feature model should be designed to represent the system variability and commonality through the result of domain analysis. We employ an ontology-based approach to incorporate the NFRs of a system into the specification of features by transforming the feature model of a software system and constructing its feature model ontology. The annotation processes is performed to enrich feature models by associating semantic metadata though using NFR ontologies; which yields how features can contribute to the satisfaction of high-level abstract objectives of the problem domain. In our use cases, feature models are annotated using the device ontology; which contributes to finding a set of configurations of software services supported by the device capabilities.

3.1.1. Feature Models for Variability Modeling

Software Product Lines (SPL) provides an effective approach for modeling variability. SPL empowers the derivation of different product family applications by reusing the realized product family assets often known as core assets. A key principle of SPL is maintaining the variation points and dependencies in order to facilitate the exploitation of commonality and the management of variability among software features. SPL practices can be utilized to support service-oriented applications to promote the reusability and adaptation of services in product families of software services. Consequently, SOA's promise of bringing reusability and loose coupling can be further augmented using SPL engineering. The features in SPL subsume generic architecture and components which are tightly coupled, whereas services in SOA are reusable loosely coupled units [4]. Treating service units in SOA as core assets of SPL allows both perspectives on reuse to be incorporated synergistically into an integrated approach. Feature modeling in SPL is a well-defined approach to capture variability and commonality of the features and allows for expressing the permissible variants and configurations of software product family. Accordingly, the large set of existing services and their shared commonalities (particularly in the domain of ubiquitous computing) makes it possible to take advantage of SOA and feature modeling in SPL engineering for modeling variability of services.

a) Feature Model

As alluded to above, the feature modeling technique is opted to represent and describe the possible configurations of system and variants in terms of features representing system functionality units. In general, there are four types of relationships related to variability concepts in the feature model. They can be classified as: *Mandatory (Required), Optional, Alternative and Or feature group*. Mandatory features must be included in the description of their parent features and presented to function as intended. Optional features may or may not be included for the basic product to function. Alternative features indicate that only one of the features from the feature groups can be used to provide that the proper feature functionality. Figure 2 depicts the graphical representation of feature models, known as feature diagram, as well as common feature diagram notation [18][5]. The sample feature model diagram, in a three-like graph structure where primitive features are leaves and compound features are interior nodes, introduces parts of the system that require

selecting and composing appropriate services for delivering particular contents based on users' requirements, functionalities of the system, and capabilities of end-point devices. The system, through conducting service adaptation, should be able to identify the capabilities of the required device, adjust the content according to device features, and select appropriate services by looking into the feature model and extracting those that have matching functionalities. Such kinds of scenarios clearly illustrate the use of SPLE in the SOA development, especially in the domain of ubiquitous computing.

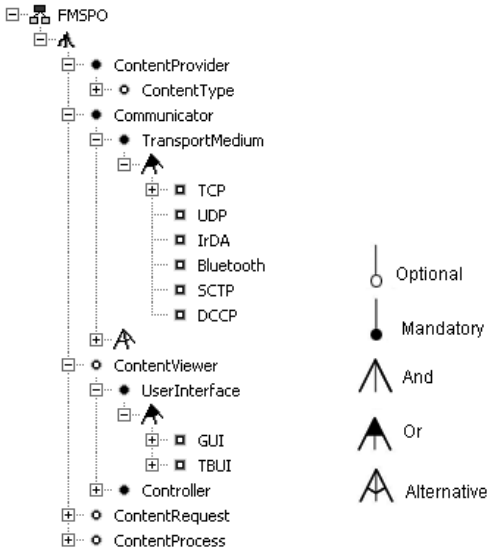

Figure 2. Feature model diagram

Feature models explicitly demonstrate different ways in which features could be composed. A valid composition of features is called a *configuration* which in turn is a valid software service product specialization. Since different applications employ features in different non-functional context, feature models help to segregate the NFRs of services from functional requirements in SOA which results in the increasing reusability of services by defining constraints among NFRs [9].Feature models enable the management of variabilities and commonalities for the set of existing services and components. Extending feature models to support non-functional or extra-functional requirements allows QoS/NFRs-driven service selection and composition. These requirements can be checked and verified against the non-functional properties interlaced with service specifications corresponding to the underlying components. In the following sections we discuss how ontologies can be exploited to extend feature models with NFRs and enable configuration of desired services based on user requests.

b) Semantic Feature Model

Current feature models mainly consider the modeling of functional features, and there is a lack of precise modeling artifacts dealing with NFRs. For instance, despite existence of several proposed approaches that bring SPL into the domain of pervasive and ubiquitous computing [13][14], most of them lack a clear specification of NFRs for service level agreement (SLA), QoS, and device capabilities [15]. Furthermore, they hardly provide possibilities for consistency check of the NFRs extended feature models, or logical reasoning over feature models which enable intelligent configuration and selection of services for product instantiation. Moreover, due to the lack of formal semantics and descriptions in a feature model, relations, dependencies and constraints of features are not specified comprehensibly through the model. Accordingly, we have employed an ontology based approach aiming at creating semantically-enabled feature models. We believe ontologies provide the appropriate means to address the aforementioned issues.

3.1.2. The Feature Model Ontology

An ontology is defined as a formal specification of a conceptualization and utilizes the representation of knowledge contained in feature models. The Feature Model Ontology (FMO) provides the semantic representation of features and relations and dependencies as well as feature constrains, which express another form of relations among features, in one model entirely. We used the approach introduced by Wang et al.[7][8] for mapping feature models to ontologies using the Web Ontology Language (OWL) [6]. The transformation of a feature model is performed through constructing mutual disjoint classes corresponding to defined nodes in the feature model and assigning a class rule constructed for individual classes created in the FMO. In other words, each class rule is associated to its corresponding feature node declaring an existential restriction and is used to define the bindings of the node's child features or to define the constraints. In our OWL model of FM, an object property is created whose asserted range is the respective feature class for the feature node. Assuming that there are i nodes in the feature model (FM), the Description Logic (DL) presentation of the above modeling can be expressed as follows:

$F_i \sqsubseteq \top$
$FM_iRule \sqsubseteq \top$
$hasF_i \sqsubseteq ObjectProperty$
$FM_iRule \equiv \exists\ hasF_i.\ F_i$
$F_i \sqsubseteq \neg F_j$, *for* $1 \le l, j \le n \wedge i \neq j$

Going back to the scenario of Section 2, let us model the need for supporting Bluetooth communication for the large display application. The DL presentation of for including Bluetooth into the feature model for our product family can be defined as follows:

Bluetooth $\sqsubseteq \top$
BluetoothRule $\sqsubseteq \top$
BluetoothRule $\equiv \exists$ *has Bluetooth.Bluetooth*,
hasBluetooth $\sqsubseteq$ *ObjectProperty*

3.1.3. Feature Model Annotation

Representation of a feature model as an ontology enables us to take advantages of ontology annotation capabilities in order to enrich the definition of domain assets with the constraints concerning the non-functional requirements of a domain. Through using expandability property of ontologies, NFRs can be formally incorporated into the ontological specification of feature models, expanded over time, or augmented with additional NFR ontologies. For a feature model to explicitly integrate NFRs to the possible set of configurations, at the design stage, it would be possible to annotate the feature model with an ontology representing NFRs of interest. That is, the ontology of service quality can be used to annotate features in the feature model with attributes such as precision, robustness, reliability, or any other relevant NFRs. We extend the FMO by defining *AnnotationProperties (ANs)* as part of our model so as to annotate the feature model using external ontologies. In essence, the ANs have feature classes as their domains and their ranges refer to the concepts (i.e., classes) in the NFRs or QoS ontologies, For the sake of the detailed analysis of the ubiquitous domain and the paper's research objective, we will use device capability ontology as a sample for describing the NFR ontology. An example of such an ontology is the W3C's Delivery Context Ontology [23]. This process and assertion of ANs' range is performed during designing the system and the existing services. The attributes are added to the feature model as part of the requirements for each feature and its corresponding service interface. Considering that we refer to the classes in FMO as FM_m and the classes in device capability ontology as DC_n, annotating FMO with the device capability ontology using an OWL object property, detonated as AN, is carried out as demonstrated in Figure 3 which shows annotation of the feature model using derived device ontology following the DL syntax as below:

AN $\sqsubseteq$ *ObjectProperty*
AN $\sqsubseteq FM_m$, *domain*(FM_m)
$\top \sqsubseteq \forall$ *AN*.DC_n, *range* (DC_n)

Considering the Bluetooth communication example, in order to relate the Bluetooth concept in the FM with the Bluetooth communication device provided by a mobile phone, the following DL associations are used:

hasBluetooth_AN $\sqsubseteq$ *ObjectProperty*
hasBluetooth_AN $\sqsubseteq$ *Bluetooth*,
domain(BluetoothRule)
$\top \sqsubseteq \forall$ *has Bluetooth_AN.BluetoothDevice*,
range (BluetoothDevice)

Due to space limits and the large collection of mappings between the device capability ontology and the feature model, we do not include a complete representation of these mappings. Every configuration instance generated from the feature model also inherits these ontology attributes with asserted values specifying the set of potential capabilities for an NFR to be satisfied.

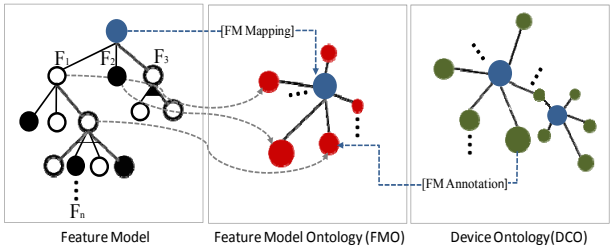

Figure 3. Feature Model Mapping and Annotation by NFRs Ontology

3.2. Service Requirement Specification

The process of selecting appropriate features for a configuration of service-oriented products is influenced by the *requests specification*, which encompasses requirements for a product configuration. In particular, we start from the definition of the *hard NFRs*. The NFRs are embedded into the requirements specification which in this case are catered by the capabilities of the target deployment platforms such as mobile devices. For example, hard NFRs describe characteristics of the device capabilities for video streaming and multimedia processing in the target deployment platform. In our approach, these hard NFP are added into an OWL knowledge base as instances of the device ontology of choice. The following ontology fragment expresses specification of a concrete delivery platform. The definition requires a particular characteristic supported by the target device display. For example, an instance of the device should be equipped with a display feature which supports specific values (i.e., *x* and *y*) for the

width and height of the display along with the component of the requested aspect ratio.

$Device \sqsubseteq \top$
$Hardware \sqsubseteq \top$
$hasHardware \sqsubseteq Device,$
$\top \sqsubseteq \forall\ hasHardware.Device$
$Display \sqsubseteq Hardware$
$\top \sqsubseteq \forall\ hasDisplay.Display$
$AspectRatio \sqsubseteq Display$
$\top \sqsubseteq \forall\ displayAspectRation.AspectRatio$
$aspectRatioHeightComponent \sqsubseteq AspectRatio$
$aspectRatioWidthComponent \sqsubseteq AspectRatio$
$\top \sqsubseteq \forall\ aspectRatioWidthComponent.int \equiv x$
$\top \sqsubseteq \forall\ aspectRatioWidthComponent.int \equiv y$
$NFR_H \geq x \quad , \quad NFR_W \geq y$

In our overall process, which is not discussed further in this paper, we also anticipate the place for the possible preference of users defined as *soft requirements*. Examples include cases like preference towards services with textual services vs. multi-media service or precision of retrieval services vs. speed of retrieval services. Soft requirements represent user preferences unlike hard requirements, which must always hold.

3.3. Software Service Products Configuration

Ontology-based modeling of the target device reflects all the capabilities of the target device. In other words, device ontology instances reflect all the NFRs as hard constraints which exclude and include the selection of services in annotated feature models. So in the process of configuration, a subset of services is selected based on hard constraints specified by device capabilities, known as *software service product specialization*. In the process of configuration, the model of the target device requires to be compliant with the annotated feature model. The features which do not satisfy the NFRs of target device are discarded pruning the feature model into a new feature model whose set of possible configurations is a subset of the original feature model. This derived feature model is referred to as a *specialization of the feature model* and the staged refinement process which constitutes *staged configuration* [17]. In terms of the OWL-based reasoning, our goal is to determine if instances of the NFRs ontology (i.e., in our case device capabilities) are consistent with respect to the annotation properties defined in the feature model. The DL-based ontology that we have discussed in our scenario comprises two knowledge bases, Feature Model Ontology ($\mathcal{FMO}$) and Device Capability Ontology ($\mathcal{DCO}$); which is denoted as $\mathcal{O} = \mathcal{FMO} \sqcup \mathcal{DCO}$. Our rule knowledge base is a quadruple $\mathcal{K} = (\mathcal{O}, \mathcal{T}, \mathcal{A}, \mathcal{R})$, where $\mathcal{T}$ is a set of concept axioms (TBox), $\mathcal{A}$ is a set of assertional axioms (Abox), and $\mathcal{R}$ is a set of rules written as inclusion axioms. Lets us provide some definition to build the ground of the feature model specialization process.

DEFINITION 1 *Let $d \in \mathcal{DCO}$ be an instance of a device which has n capabilities. Each capability for device d is an instance (i_k) of a concept (C_k) from $\mathcal{DCO}$ such that $\mathcal{A} \models C_k(i_k)$ $(1 \leq k \leq n)$. $S_{dc} = \{C_1,..,C_n\}$, represents the set of all concepts C_k from $\mathcal{DCO}$ that d supports.*

DEFINITION 2 *Let $S_{AF} \sqsubseteq \mathcal{FMO}$ be a set consisting of concepts CF_i such that $\forall CF_i.\top \mid \exists AN_i \sqsubseteq CF_i \sqcap \exists AN_i.C_k$ where $C_k \in \mathcal{DCO}$.*

The following Algorithm is introduced to specialize features from annotated feature set w.r.t hard constraints. The algorithm initially checks if each feature from S_{AF} has an annotation property AN_j (I) whose range is a class from S_{dc} (II). Those features whose properties do satisfy this condition are removed from the feature model specialization set (*SFA*). Each feature which satisfies the above conditions is further checked to see if there is at least one consistent instantiation of it in the $\mathcal{FMO}$ taking the value from $C_i(i) \in \mathcal{DCO}$ through using the annotation properties (III). Otherwise they are also removed from *SFA*.

Algorithm 3.1: Feature Model Specialization

$AN_j \sqsubseteq ObjectProperty$

$$SFA = \bigcup_{i=1}^{n} CF_i$$

for each $C_k \in S_{dc}$
 for each $CF_i \in S_{AF}$
  if $(\exists AN_j.C_j \sqcap AN_j \sqsubseteq CF_i$ *and* $C_j \in \mathcal{DCO})$ (I)
   if $(C_j \in S_{dc})$ (II)
    if $(\mathcal{A} \not\models CFi(i_j))$ (III)
     $SFA \leftarrow SFA - \{CF_i(i_j)\}$
   else
    $SFA \leftarrow SFA - \{CF_i(i_j)\}$
return (*SFA*)

Having the set of compatible features remained in the feature model; the process of composition proper of services proceeds to derive a set of suitable software service products for the requesting devices. Consistency checking and validation of a given software service configuration results from the feature model specialization and the process of the staged refinement is required to be performed in order to validate the configuration against the feature model constraints. During the process of the configuration validation, model con-

straints are checked against the derived configuration to provide proper assurance in terms of correct exclusion and inclusion of optional and mandatory features. To perform consistency checking and analysis over the OWL representation of an annotated feature model, we employ RacerPro[2] as one of the widely accepted OWL-DL reasoning engines, which supports automated class subsumption, consistency reasoning and detection of possible inconsistencies in the specialized feature model. Since some inconsistencies, derived from mutually exclusive properties (require and exclude), could not be represented by relying solely on OWL DL, due to expressivity limitations, we formulate and define a list of SWRL rules to represent all invalid states and detect conflicts or inconsistencies in the model.

3.4. Service Discovery and Specialization

The result of consistency checking in the previous step can be two folded. If the result is an inconsistent ontology (i.e., NFRs filtered out some mandatory features), we need to return to the feature analysis stage and design and refine our family of compositions to satisfy the discovered inconsistencies. Otherwise, if the result of this step is a consistent feature model specialization, we enter the process of further specialization and service discovery. The service specialization can be based on soft requirements (e.g., preferences towards certain features) of the stakeholder. Analyzing soft requirements can also be done through a similar specialization process as we have proposed for hard requirements. However, in this process, we are also considering the use of fuzzy logic [24]. Once the final configuration is obtained in which all the variability is resolved, we start the process for the final generation and deployment of the service-oriented system. We use the Web Service Modeling Ontology framework. More specifically, for the obtained feature configuration, we generate a complete description of the WSMO service compositions. This transformation is done by following the mapping rules between the feature models and abstract state machines defined in [25], while a complete implementation of the transformation is available in [26]. In our transformation, we generate all elements of the WSMO specification including, capabilities, pre- and post-conditions, transition, choreographies (along with state signature and transition rules) and orchestrations. In fact, during our project on transformations between WSMO services and feature models, we realized that in order to be able to generate complete WSMO service compositions, we need information about non-functional properties and information about ontological grounding of each feature. Therefore, our process of feature annotations, presented in this paper, perfectly complements the process of generation of complete WSMO service descriptions, We have deployed and tested the obtained services with the WSMX toolkit for discovery, mediation and composition of WSMO services [1].

[2] http://www.racer-systems.com/

4. Related Work

Wang et. al [8] provide a methodological approach to verify feature models using the Web Ontology Language. They transform feature models to ontologies by converting features to pairs of concept and rule classes with each pair presenting a feature in the feature model. This transformation coupled with constraints over the relations between the class nodes enables reasoning over the consistency of the ontology, and consequently helps with verifying the validity of the feature model and the instantiated products. Weis [19] considers pervasive applications as features that can be customized based on personal preferences. Users define configuration of applications using a graphical language with support for detailed customization at the price of increased complexity in using the language. Their approach provides a coarse approach to service composition in Pervasive environments.

One of the main issues with respect to feature models is constraining and controlling the variability of features. This is typically done by placing constraints across the feature hierarchies. These features may be represented in the form of propositional logic [18] or richer formalisms such as first order predicate logic [20] and its variants. Object Constraint Language (OCL) [21] is also used when feature models are represented in UML diagrams. Forfamel uses Weight Constraint Rule Language (WCRL) [22] to constrain the representation of features in its ontology. As a whole, the constraints over the variability enable reasoning and resolution of features depending on the configurations required for each product member or each customer of a product line.

Lee et al. [27] propose a feature-oriented product line approach for SOA which guides developers through the composition of services in a feature model. An approach to the generation of business process models in BPMN from feature models is introduced in [25]. This work inspired our transformation between feature models and WSMO service compositions.

5. Conclusions and Future works

In this paper, we have proposed a framework for development of service-oriented architectures based on the principles of multi-staged feature model specializa-

tion. The key part of the proposed approach is the use of ontologies as an underlying formalism for representation of feature models of families of service compositions. Once created, such ontology-based feature models can be easily annotated with the non-functional properties. The critical role of ontology-based reasoning takes place in the process of specialization of the annotated feature models through a set of specific requirements, which must hold in a particular service-oriented application at hand. In our concrete case, we experimented with the non-functional requirements, which specify the capabilities of the target platform for executing service compositions. The ontology-based consistency checking approach combined with the integrity constraints defined in the Semantic Web Rule Language are used in this step to discover a feature model specialization satisfying the requirements for the final system to be built (in our case those are device capabilities). Once a requested feature specialization is obtained, we go through an interactive process where further, stakeholder soft requirements (i.e., preferences) are used to obtain a feature specialization without variability. Such a specialization is then translated to a composition represented in WSMO, where the use of ontologies again plays a critical role.

In our future work, we are going to conduct an extensive experimental study to measure the effectiveness of our proposed methodology. Also, we will further explain our approach for the use of soft requirements in the process, once the hard requirements are satisfied in the service composition process. Finally, we will fully explain the process of deployment to the WSMO SOA framework.

Acknowledgement. This research was in part supported by Alberta Ingenuity through the New Faculty Award program and by Athabasca University's Mission Critical Research Fund.

6. Reference

[1] A. Haller, et al., "WSMX - a semantic service-oriented architecture," ICWS 2005., pp. 321-328 vol.1.

[2] M. P. Papazoglou, et al., "Service-Oriented Computing: State of the Art and Research Challenges," IEEE Computer, 11, 2007.

[3] P. Krogdahl, G. Luef, and C. Steindl, "Service-oriented agility: an initial analysis for the use of agile methods for SOA development," IEEE Int'l Conf. on Services Computing, 2005, pp. 93-100 vol.2.

[4] S. Ho Chang , et al., "A Variability Modeling Method for Adaptable Services in Service-Oriented Computing," In Proc. of the 11th Int'l Software Product Line Conf., 2007, pp. 261-268.

[5] K. Czarnecki, et al., "Staged configuration using feature models," In Proc. of Software Product-Line Conf. 2004, pp. 266-283.

[6] OWL, http://www.w3.org/TR/owl-features/

[7] I.Horrocks, et al., "SWRL: A Semantic Web Rule Language Combining OWL and RuleML," W3C, May 21, 2004. http://www.w3.org/Submission/SWRL

[8] H.H. Wang, et al., "Verifying feature models using OWL," Web Semantics: Science, Services and Agents on the World Wide Web, vol. 5, 2007, pp. 117-129.

[9] H. Wada, J. et al, "A Feature Modeling Support for Non-Functional Constraints in Service Oriented Architecture," IEEE International Conference on Services Computing, 2007, pp. 187-195.

[10] K.C. Kang, et al., "Feature-oriented domain analysis (FODA) feasibility study," Carnegie Mellon University, SEI 1990.

[11] K.C Kang, et al., "FORM: A feature-;oriented reuse method with domain-specific reference architectures," Annals of Software Eng., vol. 5, Jan. 1998, pp. 143-168.

[12] M.L. Griss, J. Favaro, and M. d'Alessandro, "Integrating feature modeling with the RSEB," Proc. of the 5th Int'l Conf. on Software Reuse, 1998, pp. 76-85.

[13] W.Zhang, K. M.Hensen, "Synergy Between Software Product Line and Intelligent Mobile Middleware," In Intelligent Pervasive Computing, 2007 pp. 515-520.

[14] M. Anastasopoulos, "Software Product Lines for Pervasive Computing," IESE-Report No. 044.04/E 2005.

[15] J.White, and D. C. Schmidt, "Model-Driven Product-Line Architectures for Mobile Devices," In Proc. of the 17th Ann. Conf. Int'l Fed. of Automatic Control,2008.

[16] D. Roman: Web Service Modeling Ontology. Applied Ontology, 2005, 1(1), 77–106.

[17] K. Czarnecki, S. Helsen, and U. Eisenecker, "Staged configuration using feature models," Lecture notes in computer science, 2004, pp. 266-283.

[18] D.Batory, "Feature models, grammars, and propositional formulas," In Software Product Lines Conference, LNCS 3714, pages 7–20, 2005.

[19] T. Weis, et.al, "Customizable pervasive applications," Pervasive Computing and Communications, 2006.

[20] K.Czarnecki, C.Kim, K.Kalleberg,"Feature Models are Views on Ontologies," SPLC 2006, 41-51.

[21] P.Simons.,I. Niemelä, and T.Soininen, "Extending and Implementing the Stable Model Semantics," Artificial Intelligence, 138, 2002, pp. 181-234.

[22] J.Warmer and A. Kleppe, 1999 The Object Constraint Language: Precise Modeling with UML. Addison-Wesley Longman Publishing Co., Inc.

[23] Delivery Context Ontology, W3C Working Draft, 2009, http://www.w3.org/TR/dcontology/

[24] E.Bagheri, D.Gasevic, "Feature Model Configuration using Fuzzy Propositional," to be submitted to IEEE Trans. on SMC, Part C (2009).

[25] I. Montero, J. Pena, A. Ruiz-Cortez, "From feature models to business processes," In Proc. of the 2008 IEEE Int'l Conf. on Services Computing. pp. 605-608.

[26] J.Rusk, "Transforming between WSMO services and Feature Models with ATL transformation language," http://io.acad.athabascau.ca/~jeffr/webdoc.html

[27] J. Lee, D. Muthig, M. Naab, "An approach for developing service oriented product lines," In Proc of the 12th Int'l Software Product Lines Conf. 275-284. 2008.