





# An Ontology for Service-Oriented Dynamic Software Product Lines Knowledge Management

Najla Maalaoui<sup>1</sup> <sup>a</sup>, Raoudha Beltaifa<sup>1</sup> <sup>b</sup>, Lamia Labeled Jilani<sup>1</sup> <sup>c</sup> and Raul Mazo<sup>2</sup> <sup>d</sup>

<sup>1</sup>RIADI Lab., National School of Computer Sciences, Manouba University, Tunisia

<sup>2</sup>Lab-STICC, ENSTA Bretagne, Brest, France


**Keywords:** Service-Oriented Dynamic Software Product Lines, Knowledge Management, Ontology.


**Abstract:** Service oriented dynamic software product line (SO-DSPL) engineering provides a development paradigm for building configurable service-oriented applications and adapting them at runtime according to their context. To support context-aware and runtime adaptation of services, SO-DSPLs consider various aspects of knowledge, such as users' contexts, product lines properties, services description and QoS. In fact, the knowledge variety produces variability that must be taken into account. Thus, wealth and diversity of this knowledge require an efficient knowledge management (KM) tool to ensure knowledge-based SO-DSPL engineering. The main challenge is to overcome the lack of a multi-dimensional and unified conceptualization of knowledge. Practically, most of existing knowledge representations consider particular dimensions that depend on specific product lines, ignore the semantic between them and do not handle knowledge variability. To tackle this challenge, we propose in this paper an ontology for SO-DSPL knowledge management that establishes a common conceptualization about SO-DSPLs dimensions and provides a unified and sharable knowledge base. This ontology can serve several SO-DSPL activities and KM-related purposes, such as defining a common vocabulary for knowledge workers with respect to the SO-DSPL domain, structuring SO-DSPL knowledge repositories, and providing a knowledge base to configure and recommend services to be used for the building and dynamic adaptation of configurable service-oriented applications. In this paper, we present the ontology dimensions by means of sub-ontologies in order to promote their reuse.


## 1 INTRODUCTION


Service oriented architecture (SOA) provides a promising mechanism for supporting continuously changing customers' needs, context and expectations, as more sophisticated software systems are connected to the Internet. With the variability of user needs and context, developing reusable and dynamically reconfigurable service-based systems that can combine services in various configurations and tailored to meet different customers' needs and contexts became a main challenge. To tackle this challenge, Service Oriented Dynamic Product Lines (SO-DSPL) Engineering is addressed by combining SOA with Dynamic Software Product Lines Engineering (DSPL) in order to achieve the development of more flexible

and customized service-oriented applications (Capilla et al., 2014). The fusion of these two popular modeling paradigms provides a great potential to develop service-based solutions that can tackle various challenges in development and infrastructure management of service-oriented systems. To deliver on these promises, SO-DSPLs overcome a number of software engineering challenges. 1) Ensure runtime variability by the activation and deactivation of features. 2) Adapting system functionalities to a new context. 3) Considering context-aware and self-adaptation properties to handle changes in system variants dynamically. The complexity and the importance of the overcome challenges makes the knowledge to be considered more diverse, rich and carries semantics whose exploitation and management can serve different activities of the product line(PL). To tackle the mentioned challenges, SO-DSPL considers a variety of knowledge including user context and requirements, PL variability, services knowledge and service-oriented applications adaptation knowledge.

<sup>a</sup>  <https://orcid.org/0000-0002-3896-920X>

<sup>b</sup>  <https://orcid.org/0000-0003-4096-5010>

<sup>c</sup>  <https://orcid.org/0000-0001-7842-0185>

<sup>d</sup>  <https://orcid.org/0000-0003-0629-1542>

Therefore, Knowledge Management (KM) emerges as an important means to manage SO-DSPL knowledge since KM principles can help capturing and representing SO-DSPL knowledge in a manageable way. In this context, representing and sharing knowledge by using ontologies have emerged as a promising means to minimize ambiguity, unify and conceptualize knowledge. With the lack of a unified SO-DSPL knowledge conceptualization, we tackle the aforementioned challenges by proposing an ontology for SO-DSPL knowledge management. Our ontology aims to conceptualize, unify and emphasize the semantics carried by the different SO-DSPL knowledge in a sharable, exploitable and reusable way. As well, our proposal focuses on knowledge variability which must be considered in the context of SO-DSPL framework. In order to promote reusability, our proposed ontology is structured by sub-ontologies.

The rest of the paper is organized as follows: Section 2 introduces SO-DSPL engineering with a brief overview. Section 3 presents the construction of the ontology. Ontology implementation is presented in section 4. Section 5 reports the evaluation of the proposed ontology. Related works are presented in Section 6. Finally, Section 7 concludes the paper and describes future works.

## 2 SERVICE-ORIENTED DYNAMIC SOFTWARE PRODUCT LINES

Software product line engineering is a software reuse paradigm that aims to develop a family of products with reduced time to market and improved quality (Capilla et al., 2014). Within the SPLE field, dynamic software product lines (DSPLs) have emerged as a promising means to develop reusable and dynamically reconfigurable core assets. Service-oriented dynamic software product lines (SO-DSPL) represent a class of DSPLs that are built on services and service-oriented architectures (SOAs) (Capilla et al., 2014). SO-DSPL support user needs and expectations in a continuously changing environment. SO-DSPL combine services in various configurations and contexts, simplifying the deployment of product variants that are mostly based on the same core services but tailored to meet different customers' needs. Feature modeling is the main activity to represent and manage PL requirements as reusable assets by allowing users to derive customized product configurations. Product configuration refers to the decision-making process of selecting an optimal set of features from the PL that

comply with the feature model (FM) constraints. FMs are hierarchical models that capture the commonality and variability of a PL by defining its features, their dependencies and constraints between them. Figure 1 shows an example of a feature model associated to a Smart Home PL.

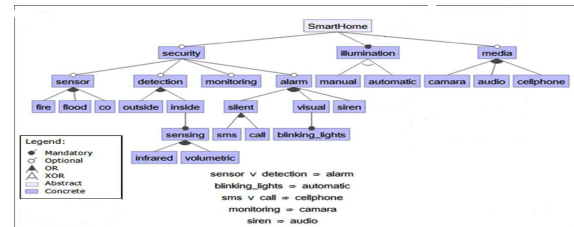


Figure 1: Extract of Smart home feature model.

## 3 OntoSO-DSPL: ONTOLOGY FOR SO-DSPL

This section presents an ontology for Knowledge management of SO-DSPL to be used in SO-DSPL activities and related purposes. Our proposed ontology is responsible for structuring, unifying, reasoning, disseminating SO-DSPL data and to be used in SO-DSPL activities such as services recommendation and selection, dynamic adaptation, runtime variability management and SO-DSPL context reasoning. In addition, our ontology includes swrl rules that are responsible for inferring new knowledge and providing new facts through its reasoning capabilities. The method for constructing our proposed ontology is adapted from ontology construction methods proposed by (Fernández-López et al., 1997). The construction process contains five main steps: objective, knowledge acquisition, conceptualization, implementation and validation.

### 3.1 Objective

The main objective of the target ontology is to provide a knowledge capitalization in SO-DSPL framework. Our proposed ontology aims to describe common conceptualization of useful knowledge including concepts considered in a SO-DSPL framework and knowledge related to user context and requirements, PL, services and dynamic adaptation. Thus, we address to conceptualize a unified model for a SO-DSPL from different dimensions. Our proposed ontology should harmonize the SO-DSPL terminology and help engineers and researchers to configure products, build and propose approaches that address the SO-DSPL's activities. Our proposed ontology is de-

signed as a core conceptual model to be used for several purposes, such as: 1) A reference model for annotating SO-DSPL knowledge in a semantic documentation approach, 2) To support human-learning about key concepts related to SO-DSPLE, 3) To be used by SO-DSPL activities and 4) To Reuse and integrate proposed sub-ontologies independently.

### 3.2 Knowledge Acquisition

The ontology that we propose is multi-dimensional. Its dimensions represent the SO-DSPL areas for which various ontologies, models and frameworks was defined in literature. These areas are SO-DSPL, dynamic adaptation, web service (WS) modeling and user context modeling. In our work, we analyzed these works and then we defined a common knowledge for each area based on advices given in (Fernández-López et al., 1997). In the following, we present the knowledge acquisition resources. For SO-DSPL area, knowledge are extracted from (Capilla et al., 2014) and (Bashari et al., 2017), which provide a more comprehensive treatment of DSPL models, properties, challenges and their solution architectures. In addition, we have exploited ontologies dealing with semantic DSPL variability modeling (Dehmouch et al., 2019). As well, we have exploited the work (Alfárez et al., 2014) to extract knowledge according to the dynamic adaptation of service composition using SO-DSPL. Accordingly, dynamic, runtime and self-adaptation knowledge are acquired from: (Andersson et al., 2007) that define MAPE-K loop model, (Sabatucci et al., 2018) that presents self-adaptive systems meta-model, (Jaroucheh et al., 2010) that considers the context in feature selection activity and (Guinea et al., 2012) that presents a set of dimensions which explain how runtime adaptation could be realized using DSPLE. To acquire WS knowledge, we are based on standards and ontologies focusing on WS description. Thus, we are based on previous literature surveys (Li et al., 2006) to select relevant works. For instance, we have analyzed the basic WSDL standard developed by IBM and Microsoft, namely WSDL and its semantic annotated extensions, such as WSDL-S and SAWSDL (Battle, 2007). In addition, we have analyzed the semantic markup of WS approaches that address the service description with the consideration of semantic. The analyzed proposals include WSMO, OWL-S, SWSA and SWSL(Battle, 2005). Due to the semantic expressivity richness in OWL-S and SAWSDL, we have decided to use terminologies, concepts and terms extracted from these two most used standards to conceptualize our proposed WS Sub-ontology. As well, we have focused on ac-

quiring SLA knowledge that plays an important role in WS selection activity. Thus, we have considered in our work knowledge extracted from USDL standard, WSLA Framework and the PaaS Level SLA Description Language presented in (Hofman et al., 2020). To acquire context knowledge, we have reused different context ontologies. We analyzed about ten context ontologies such as (Fensel et al., 2010; Zhong-Jun et al., 2016; Aguilar et al., 2018; Brickley and Miller, 2014), based on previous literature surveys. From the mentioned resources, relevant concepts and relationships were extracted through their systematic and syntactic analysis and terms are unified using "WordNet" framework.

### 3.3 Conceptualization

The conceptual modeling of the proposed sub-ontologies is presented as an OntoUML meta-model (Guizzardi, 2005). The meta-model classes correspond to classes of our ontology; the dependencies between meta-model classes are represented as ontology object properties; and the attributes of meta-model classes are represented as ontology datatype properties. Since SO-DSPL domain is complex, our proposed ontology (OntoSO-DSPL) was developed in a modular way. OntoSO-DSPL has four modules (sub-ontologies) defined as: User context sub-ontology, service sub-ontology, DSPL sub-ontology and adaptation sub-ontology where each of them is interested in covering a particular dimension.

**User Context Sub-ontology Meta-model.** This sub-ontology meta-model (see Figure2) consolidates and unifies relevant user context knowledge in one hand, and emphasizes context and requirement variability in the other hand.

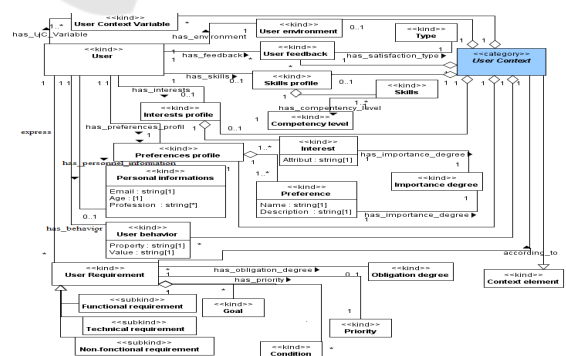


Figure 2: User context sub-Ontology meta-model.

The user context is defined by Preferences profile (PP), Skills profile (SP), Interests profile (IP), User behavior (UB), User feedback (UF), Personal information (PI), User environment (UE) and User context

variable (UCv). “PP” represents soft requirements provided by users that reflect a favored wish from the final derived application. “SP” represents competencies and abilities denoted by the user. “IP” denotes the subjects that a user wants. “UB” represents the behavior of the user during the use of an application. It can be represented by the number of its use. “UF” represents user’s reactions and opinions about an application or a service. “UE” presents information about the surrounding environment such as user location or weather. Variability modeling of the user context is considered in our ontology. For example, preferences can be expressed by a natural language or by rating SO-DSPL options. As well, the specified preferences can be mandatory or optional. Thus, we assign to user preferences an “Importance degree”, which can be “low”, “high” or “medium”. As well, to give more semantic to the representation of user preferences, we offer the possibility of specifying dependencies between preferences by defining the semantic relations “and” and “or” between them. In addition, we address interests and skills variability by assigning to interests an “Importance degree” (“low”, “high” or “medium”) and to skills a “Competency level” (beginner and expert). In addition, “UF” can be positive or negative, and it concerns a SO-DSPL asset (feature, service or goal). Since contextual information is difficult to count because each domain has its own context properties, we assign to the user context variables that represent contextual information related to the domain of the used PL. For example, these variables can represent biometric and biological data, which is needed for medical context-aware applications. The success of the product derivation activity in the PL is based on the satisfaction of user requirements, which carries a semantics allowing the infer of new knowledge that promote their satisfaction. For this purpose, we have integrated the user requirements knowledge to the user’s context. Since users express their needs by various requirements with different importance degrees, our ontology represents their variability. In fact, a user requirement (UR) may be a “Functional requirement” (FR), a “Non-functional requirement” (NFR) or a “Technical requirement”. A “Functional requirement” specifies the operations and activities that the derived application must be able to perform. Based on our running example, a “FR” FR1 can be specified by a user as follows: ‘the sensor fire shall trigger the alarm when a fire is detected’. A “Non-functional requirement” describes the application’s operation capabilities, properties and constraints required by the end-user. Based on our running example, an example of a “NFR” requirement is: the sensor fire has a response

time that does not exceeds 0.02 milliseconds’. In turn, the “UR” may contain conditions that describe constraints or properties. For example, in the “FR” “FR1” presented above, the condition is “when a fire is detected”. As well, “UR” have “Goals” that must be achieved by the derived application. As an example, the accorded goal to the “FR” “FR1” is underlined. A user may express his requirement’s importance by using an obligation degree and by assigning a priority to it. The priority indicates that the “UR” is “Essential”, “Recommended” or “Desirable”. “Essential” UR indicates that the requirement must be achieved, while a “Recommended” one is advisable to be achieved. However, Desirable “UR” is not necessarily achievable. An obligation degree represents a term used to add a precision to the “UR”. The term can be “shall”, “should”, “could”, “must” or “may”.

**DSPL Sub-ontology Meta-model.** This section presents the DSPL sub-ontology meta-model (see Figure3), which is interested in conceptualizing DSPLs variability expressed through FMs in one hand and in modeling the possible interactions between the DSPL knowledge, contextual data and its derived configurations in the other hand. As well, it focuses on reducing the lack of expressiveness of FMs in DSPL framework.

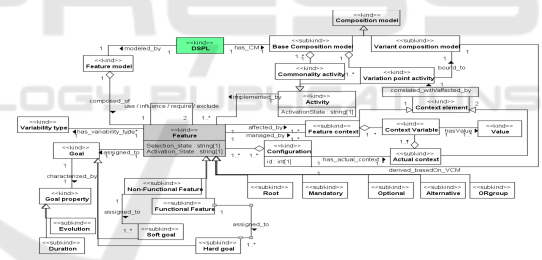


Figure 3: DSPL sub-ontology meta-model.

As an ontological representation, the “Feature model” is composed of “Features” that can be “Mandatory” or “Optional”. Its dependency with its parent can be “Alternative” or “ORgroup”. A feature model has a root feature. Besides, each two features can be related with a “Relation”. Two types of relations are identified: “feature\_constraint” and “behavioral\_relation”. A “feature\_constraint” can be “require” or “exclude” constraint. The “behavioral\_relation” can be divided into “use” and “influence” relations. A “use” relation denotes that a feature uses another one to accomplish its job and the “influence” relation indicates that the behavior of a feature has an impact on the behavior of the related feature. A “Feature” can be divided into “Functional feature” (e.g.: illumination) and “Non-functional feature” (e.g.: response time). Besides, a DSPL promote

variability management according to the actual context, thus, a “Feature” can have a variability type that can be “Static” or “Dynamic”. Static variability is managed during product derivation process while a dynamic variability is managed at runtime based on the current context. In this sense, the property “selection state” of the feature indicates whether the feature belongs or not to a particular product. This property presents the following range of values: “eliminated”, “selected”. Thus, a set of selected features represents a configuration that turn in its actual context (eg. ”Runny weather” context in the case of Smart home configuration). The dynamic reconfiguration/adaptation of a configuration, is managed by the property “activation state” of its features. This property can be represented by the values: active and inactive. The selection/activation of features is managed by the feature context that denote the condition that must be true to take decision. A “Feature” realizes one or more Goals and a “Goal”, which can be Hard or Soft. Based on our running example, a hard goal can be “detect fire in the smart home” and a soft goal can be “fire detection very fast”. In the DSPL framework, a “Goal” is characterized by its ”Duration” and ”Evolution” properties. These two properties indicate respectively whether the goal can change within the lifetime of the system and its validity through the system lifetime. In order to promote the automatic derivation of the configuration, a base composition model which is composed of activities is assigned to the DSPL to describes its business processes (Alférez et al., 2014).

**Service Sub-ontology Meta-model.** Our proposed service sub-ontology (see Figure 4) aims to unify WS description in a SO-DSPL framework in one hand, and to focus on WS variability introduced by the SO-DSPL engineering.

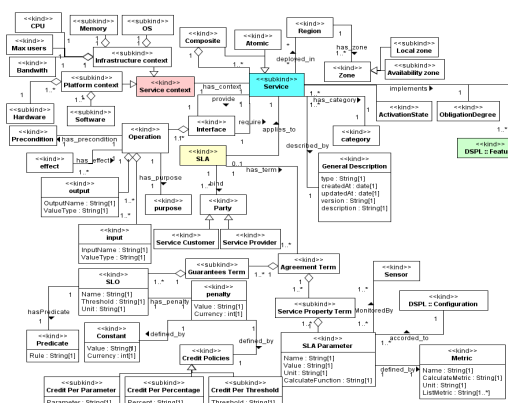


Figure 4: Service sub-ontology meta-model.

A service is identified by its general description.

In addition, it is categorized by its domain of interest (“Category”), i.e. security, finance, etc. A service is deployed in a “Region”, which is defined as a cluster of data centers. Each region has one or more availability zone, and local zone. An Availability Zone consists of one or more data centers inside a region while a “Local Zone” is not inside a “Region” boundary and places closer to our end-users. A service can be atomic or composite. A service interface encapsulates operations provided by the service. It is characterized by input, output, precondition and post condition that can be described by SWRL rules. Each operation is semantically described through its purposes that describe its business functionalities. As well, a service “requires” one or more other services interfaces to accomplish its tasks. A service is executed in its own context. This latter is divided into ” Platform context” that denote the hardware and the software which the service is executed on, and ”Infrastructure context” which denotes its infrastructure characteristics. The service sub-ontology takes intention of SLA which play an important role in service related activities. The SLA binds between two parties presented by service provider and service customer. Each SLA contains the concept “Agreement Term” which is composed of “Service Property Term” and “Guarantees Terms” which composed of “SLO”. The “SLA parameter” represents the QoS to calculate that are defined by “CalculateFunction” in order to measure the SLA parameters values (e.g. Response Time). The SLA parameter is defined by metrics, “CalculateMetric”. The “SLO” represents the service level objectives that must be respected by the provider. It has a “Predicate” that Specifies what is promised by the service provider to its consumer. In addition, for each SLA violation, a policies are used to calculate the required penalty value. For example, if the response time value of the WS exceeds the threshold defined in the SLO, a penalty is charged to the provider. According to the SO-DSPL framework, a service can be activated for a feature and deactivated for another, as well, it can be mandatory for a feature and optional for another. This variety introduces service variability. Thus, we specify service variability by assigning “Activation State” and “Obligation degree” to the service “according to” the feature. “Activation State” indicates that the service will be used or not to satisfy feature goals, it only receive the values: “Active” or ‘inactive’. A service ”Obligation degree” receives the values “Mandatory” or “Optional”.

**Adaptation Sub-ontology Meta-model.** SO-DSPLs provides a mechanism to automatically adapt products at runtime due to changes of the system, its requirements, or the environment in which they are

ployed. In this section, adaptation sub-ontology meta-model (see Figure5) conceptualizes various knowledge to characterize the adaptation of PL products.

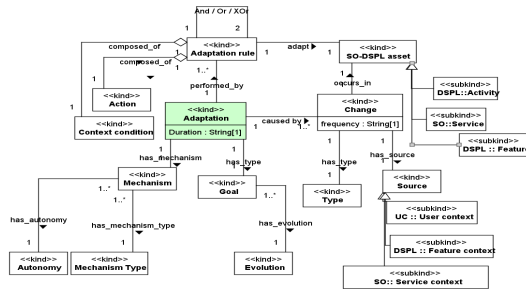


Figure 5: Adaptation sub-ontology meta-model.

An adaptation is caused by a change that can be produced by different sources such as: (i) user context, (ii) feature context and (iii) service context. An adaptation change can be functional (e.g., user requirement change), non-functional (e.g., QOS change) or technological (e.g., service execution environment). Besides, an adaptation change is characterized by a frequency of its occurrence that presents the following range of values: “frequent”, “rare”. Frequent indicates that adaptation change is frequently occurred in the current application and ‘rare’ denotes the opposite case. In the context of SO-DSPL, two types of adaptation mechanism can be performed: Architectural adaptation or service adaptation. The architectural adaptation refers to the structure or the architecture changes to modify the collaboration between services or the incorporation of new ones. The service adaptation refers to substitution of a service with another one. To indicate how long the adaptation lasts, a duration property characterizes it. An adaptation is performed by adaptation rules. The adaptation rule is composed of a condition that triggers the adaptation, an action that presents the following range of values: “active”, “inactive” and “substitute” and a DSPL asset in which the action is performed. As an example of an adaptation rule : ‘if the availability of the service S1 is less than 0,3, substitute S1 with S2. “if the availability of the service S1 is less than 0,3” is the condition, “substitute” is the action and “S1” in the service. In order to present composite adaptation rules, semantic relation “and”, “or” and “xor” are defined to relate them. The adaptation is characterized by its autonomy that can be Manual or Automatic. An adaptation is triggered for different goals such as Self-configuring, Self-optimizing and Self-healing goals.

### 3.3.1 OntoSO-DSPL Rules

Based on the semantic and the relationship expressed by SO-DSPL sub-ontologies, new knowledge can be

automatically inferred by the SWRL rules. In table 1, we present some examples of the defined rules.

## 4 ONTOLOGY IMPLEMENTATION

Among the different editors of ontologies, Protege (version 5.2.0) (Horridge et al., 2004) was chosen since it is an extensible environment for creating, checking constraints, and extracting ontologies and knowledge bases. OWL 1.0 (Web Ontology Language) was used for the development of OntoSO-DSPL as recommended by the World Wide Web Consortium (W3C). For the creation of inference rules, SWRL (Semantic Web Rule Language) was used. In the following section, in order to check the ability of OntoSO-DSPL to represent concrete SO-DSPL of the real world, we instantiated its concepts and relations to represent “smart home SO-DSPL” knowledge. Figure 6 represents an extract of the DSPL sub-ontology instantiation for Smart home SO-DSPL. As it can be seen in the figure 6, a “smartHome” DSPL, its according feature model and features with different kinds are instantiated. Indeed, instances of feature context (Sunny,Rainy) are accorded to the features “Open, Close” and some goals are assigned to the instantiated features. Based on the instantiated features, a configuration (Conf1) is derived and its actual context “WeatherContext” is assigned.

## 5 ONTOLOGY EVALUATION

Several ontology evaluation approaches have been proposed. Studying the most common evaluation approaches has conducted us to choose the criteria-based evaluation approach (Jonathan et al., 2005). This approach consists on using a set of criteria to verify the ontology. We selected a list of criteria among those which are outlined in the work of Yu et al.(Jonathan et al., 2005). These criteria are: consistency, correctness and usability.

**Evaluating Correctness and Consistency:** To carry out this validation we rely on the advices given in (Stuckenschmidt et al., 2009) that specifies that “a basic requirement for a modular ontology to be correct is that each module is correct”. This is the first perspective that we have adopted to conduct the correctness validation of the proposed ontology. Second, we have conducted a syntactic correctness, consistency, and consistency between instances. For this purpose, we have used 1) the Pellet reasoning engine (Hor-

Table 1: SO-DSPL SWRL rules.

ID	SO-DSPL SWRL rules
R1	Requirement(?r) ∧ swrlb:contains (?r, “ Should ”)-> has_priority(?r,Recommended): <i>If the requirement contains the term ‘Should’ then its priority is ‘Recommended’</i>
R2	Goal (?g) ∧ characterized_by(?g,Temporary_duration) ∧ Feature(?f) ∧ assigned_to(?f,?g)-> has_variability_type(?f, Dynamic): <i>If the goal is assigned to a feature and characterized by a temporary duration then the feature has a dynamic variability.</i>
R3	Requirement(?r) ∧ Goal(?g) ∧ Feature(?f) ∧ satisfy(?g,r) ∧ assigned_to(?f,?g) -> selection_state (?f,“ selected ”): <i>If a goal satisfy a user requirement then the feature assigned to the requirement has a ”selected” state</i>

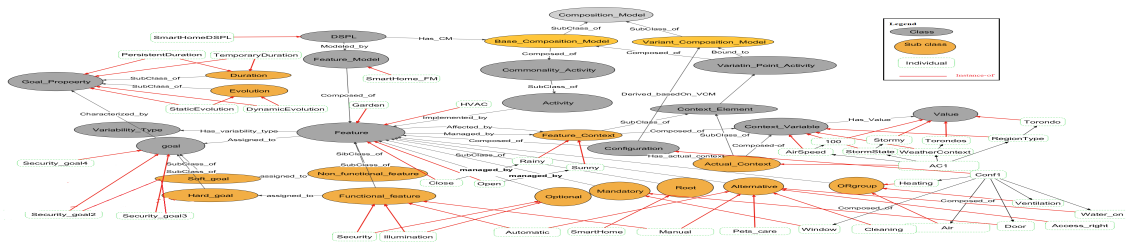


Figure 6: Extract of smart home DSPL sub-ontology.

ridge et al., 2004) 2) RaDON (Ji et al., 2009) plugging to verify inconsistencies and incoherencies in ontologies. In this validation process, the obtained results showed that none inconsistency were found in any of the modules of the proposed ontology.

**Usage:** In this section, we validate the usage of the proposed ontology in SO-DSPL activities as we have mentioned in section 3 by triggering reasoning over the ontology. Specifically, to validate OntoSO-DSPL usage, we illustrate its exploitation in the dynamic adaptation activity in a smart home DSPL.

**OntoSO-DSPL Dynamic Adaptation:** The reasoning capabilities of our proposed OntoSO-DSPL in the case of dynamic adaptation are defined as the ability of deducing new knowledge and triggering a dynamic adaptation of the smart home configuration according to its actual context.

**Triggering Dynamic Adaptation Rule:**

```

DSPL(SmartHome)'Configuration'(cf)'hasConfiguration'(SmartHome,?cf)'ActualContext'(?ac)'has_context'(?cL?ac)'ContextCondition'(cCon)'swrlb:equals'(?ac,?cCon)'AdaptationRule'(?ar)'composed_of'(?ar,Activation)'composed_of'(?ar,?cCon)'Feature'(f)'adapt'(?ar,?f)->Activation_State('Activated')'composed_of'(cf,?f)'Selection_State('Selected')
    
```

Using the rule "Triggering Dynamic Adaptation", the dynamic adaptation of a configuration can be triggered based on the actual context that must be equal to the context condition of the adaptation rule in order to activate, deactivate, select or deselect a feature.

## 6 RELATED WORKS

In this section, we present the state of the art related to the modeling of SO-DSPL, DSPL, context, self-adaptation and services. Knowledge management for SO-DSPL are not yet devoted. However, different works are interested in the considered SO-DSPL dimensions areas. Regarding SO-DSPL knowledge description, reference framework and standards are addressed to this purpose. Capilla et al. (Capilla et al., 2014) present an overview of DSPL architectures, properties, challenges and its application domains such as SO-DSPL. As well, different works are focusing on self-adaptive systems modeling. In (Sabatucci et al., 2018), a unified metamodel of four types of adaptation is proposed to smart systems. Concerning WS description, many standards and approaches have been proposed. IBM and Microsoft have developed a standard based on XML format to describe WS, namely WSDL (Battle, 2007), which is the most used WS description language until today. With the lacks semantic expressivity in WSDL, WSDL-S and SAWSDL(Battle, 2007) are proposed to extend WSDL with semantic annotations. In addition, semantic markup of WS has been proposed to address the service description such as OWL-S (Martin, ). Focusing on context modeling, there are several works that have developed the modeling of context through ontologies. In (Brickley and Miller, 2014), FOAF ontology is presented for describing people and the relationships between them. In (Fensel et al., 2010), an ontological modeling of the user profile is proposed.

In (Zhong-Jun et al., 2016), the authors present a Meta Context Ontology Model (MCOnt), where the context is divided in three categories: the internal context, the external context and the boundary context. In (Aguilar et al., 2018), the authors present a context awareness meta ontology modeling called “CAME-Onto”, which groups six contextual classes (user, service, location, activity, time and device). Based on the studied works, we conclude that SO-DSPL management and conceptualization knowledge are not yet addressed. As well, considered dimensions are modeled undependably which make the relationship between them absolutely absent. Thus, we propose an ontology for SO-DSPL framework in order to conceptualize and unify the considered knowledge.

## 7 CONCLUSION

In this paper, we have proposed a SO-DSPL knowledge management ontology, called OntoSO-DSPL. This latter highlights relationships between different dimensions in a SO-DSPL framework as knowledge related to user context and requirements, PL, services and dynamic adaption. The main objective of the target ontology is to provide a knowledge capitalization. Compared to existing ontologies for DSPL, our proposed one contains all the knowledge necessary in several activities related to SO-DSPL framework, such as contextual services recommendation and dynamic adaptation of recommended service-oriented applications. As perspectives for the near future work, we are going to add semantics to all the meta-models (like OCL). Also, we will use the conceptualized knowledge to evaluate the performance of OntoSO-DSPL in different domains related to real cases.

## REFERENCES

- Aguilar, J., Jerez, M., and Rodríguez, T. (2018). Cameonto: Context awareness meta ontology modeling. *Applied Computing and Informatics*, 14(2):202 – 213.
- Alfárez, G., Pelechano, V., Mazo, R., Salinesi, C., and Diaz, D. (2014). Dynamic adaptation of service compositions with variability models. *Journal of Systems and Software*, 91:24 – 47.
- Andersson, J., de Lemos, R., Malek, S., and Weyns, D. (2007). *Modeling Dimensions of Self-Adaptive Software Systems*, volume 5525. Springer International Publishing, Berlin, Heidelberg.
- Bashari, M., Bagheri, E., and W.Du (2017). Dynamic software product line engineering: A reference framework. *International Journal of Software Engineering and Knowledge Engineering*, pages 191–234.
- Battle, S. (2005). SWSF. <https://www.w3.org/Submission/SWSF-SWSL/>. [Online; accessed 19-mars-2020].
- Battle, S. (2007). SAWDSL. <https://www.w3.org/TR/sawddl/>. [Online; accessed 19-mars-2020].
- Brickley, D. and Miller, L. (2014). FOAF ontology. <http://xmlns.com/foaf/spec/>. [Online; accessed 19-mars-2020].
- Capilla, R., Bosch, J., Trinidad, P., Ruiz-Cortes, A., and Hinchey, M. (2014). Overview of dynamic software product line architectures and techniques: observations from research and industry. *The Journal of Systems and Software*, pages 3–23.
- Dehmouch, I., Asri, B., Rhanoui, M., and Elmaallam, M. (2019). Feature models preconfiguration based on user profiling. *Comput. Inf.*, 12:59–71.
- Fensel, D., Lausen, H., Polleres, A., Bruijn, J. D., Stollberg, M., Roman, D., and Domingue, J. (2010). *Enabling Semantic Web Services*. Springer, first ed. edition.
- Fernández-López, M., Gómez-Pérez, A., and Juristo, N. M. (1997). From ontological art towards ontological engineering. In: *Proceedings of the Ontological Engineering AAAI-*, 97:33–40.
- Guinea, S., Baresi, L., and Pasquale, L. (2012). Service-oriented dynamic software product lines. *Computer*, 45(10):42–48.
- Guizzardi, G. (2005). OntoUML. <https://ontouml.org/>. [Online; accessed 19-January-2020].
- Hofman, C., , and Roubtsova, E. (2020). A reference model for a service level agreement. In *Business Modeling and Software Design*, pages 55–68, Cham. Springer International Publishing.
- Horridge, M., Knublauch, H., Rector, A., Stevens, R., and Wroe, C. (2004). *A Practical Guide To Building OWL Ontologies Using The Protégé-OWL Plugin and CO-ODE Tools Edition 1.0*. University of Manchester.
- Jaroucheh, Z., Liu, X., and Smith, S. (2010). Candel: Product line based dynamic context management for pervasive applications. In *2010 International Conference on Complex, Intelligent and Software Intensive Systems*, pages 209–216.
- Ji, Q., Haase, P., Qi, G., Hitzler, P., and Stadtmüller, S. (2009). Radon — repair and diagnosis in ontology networks. In Aroyo, L., Traverso, P., Ciravegna, F., Cimiano, P., Heath, T., Hyvönen, E., Mizoguchi, R., Oren, E., Sabou, M., , and Simperl, E., editors, *The Semantic Web: Research and Applications*, pages 863–867, Berlin, Heidelberg. Springer Berlin Heidelberg.
- Jonathan, Y., James, T., and Audrey, T. (2005). Evaluating ontology criteria for requirements in a geographic travel domain. In Meersman, Robert, Tari, and Zahir, editors, *On the Move to Meaningful Internet Systems 2005: CoopIS, DOA, and ODBASE*, pages 1517–1534, Berlin, Heidelberg. Springer Berlin Heidelberg.
- Li, K., Verma, K., Mulye, R., Rabbani, R., Miller, J., and Sheth, A. (2006). *Designing Semantic Web Processes: The Wsdl-S Approach*, pages 161–193. Springer US, Boston, MA.
- Martin, D. OWL-S. <https://www.w3.org/Submission/OWL-S/>. [Online; accessed 19-mars-2020].



- Sabatucci, L., Seidita, V., , and Cossentino, M. (2018). The four types of self-adaptive systems: A metamodel. In Pietro, G. D., Gallo, L., Howlett, R., and Jain, L., editors, *Intelligent Interactive Multimedia Systems and Services 2017*, pages 440–450, Cham. Springer International Publishing.
- Stuckenschmidt, H., Parent, C., and Spaccapietra, S. (2009). *concepts, theories and techniques for knowledge modularization*. Springer.
- Zhong-Jun, L., Guan-Yu, L., and Ying, P. (2016). A method of meta-context ontology modeling and uncertainty reasoning in swot. In *International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery*, pages 128— 135, china. IEEE.

