A Proposal for a Comparison Framework for the Management of Variability in Web Services Composition

Yassine Jamoussi[1,2 *], Sabrine Bahri [2]

[1] Sultan Qaboos University, Department of Computer Science, Muscat, Oman.
[2] Manouba University / RIADI Laboratory, Tunis, Tunisia.

* Corresponding author. Tel.: +968 2414 2464; email: yessine@squ.edu.om
Manuscript submitted December 10, 2016; accepted May, 2017.
doi: 10.17706/jsw.12.4.303-314

Abstract: Service Oriented Architecture (SOA) is currently considered as one of the most advanced means to realize quickly modular and flexible applications. A key concern for flexibility in SOA is the variability. Several works on variability were conducted. However, few studies have proposed classifications on the variability in the composition of web services. In this article, we propose an original approach to compare the management of variability in web services composition. This approach is based on a framework that considers the variability according to four dimensions namely the form, content, purpose, and life cycle. Each dimension is defined by a set of attributes that characterizes a particular facet of the variability.

Key words: Variability, SOA, Web services composition, comparison framework.

1. Introduction

The contemporary mutations are so fast and become a success factor for companies. In order to accommodate quickly with the perpetually changing needs of these companies, the Service-Oriented Architecture (SOA) has emerged. SOA is currently considered as one of the most advanced means to realize quickly integrated, modular and flexible applications. A key concern for flexibility in SOA is the variability [1].

Many approaches have been interested to manage the variability [2], [3]. They are mainly built on different models and technical aspects. Despite they seems to be simple, it is difficult to select the appropriate approach that can be applied for a specific context.

Recently, with the increasing number of the approaches considering the management of variability, some studies were conducted in order to compare these approaches. Among those are, for example, the studies conducted by (Valença et al., 2013) [4] and (Galster et al., 2014) [5] in which the authors focused to understand the variability at the business processes. In (Rusli et al., 2011)[6], (Czarnecki et al., 2012) [7] and (Mahdavi et al., 2013)[8] the authors evaluated the quality of the variability in service-based systems. In general, these studies are based on a set of questions to seek and clarify a specific aspect of the variability.

We noticed that little attention has been paid to understand the variability ensuring the flexibility in SOA. However, we do believe that a good understanding of the different facets of variability will open new perspectives for improving and exploiting research results at best.

In this paper, we propose an original approach for comparing web services composition approaches that are managing the variability. This approach differs from existing comparative approaches by the fact that it provides a framework for comparison. The proposed framework is inspired from the comparative framework developed

for the study and the understanding of the requirement-engineering approaches (Rolland et al.) [9]. Thus, in our work we look at the existing variability management approaches according to different dimensions. Each dimension is defined by a set of attributes that characterizes a particular facet of the variability. The considered facets are organized around the following questions:

- In what form is expressed the variability?
- What knowledge is expressed in the variability?
- Why do we use the variability?
- How to handle the variability?

The remainder of the paper is organized as follows. Section 2 presents generalities about variability. Section 3 details the proposed comparison framework. Section 4 reveals the use of the framework by illustrating its use for the understanding and comparison of the most relevant approaches that are managing the variability in SOA. Finally, Section 5 concludes the work.

2. Generalities on Variability

We mainly present the notion of variability followed by variability in SOA.

2.1. Variability

Variability is the ability of a software artifact or software system to be extended, customized or configured to be (re-) used in a specific context [10]. Variability specifies parts of the system and its architecture that remain variable and that are not fully defined during early design [11], [12].

Many types of today's software systems are built through variability, e.g., Software Product Line (SPL) [13], self-adaptive systems, open platforms, or service-based systems (variability is used depending on the nature of the service, which can be atomic or composite).

Variation points, i.e., the places where the changes may occur, usually define the variability. Thus, it is preoccupied by the following definition i) what parts called "variation points" can vary in time or in a domain space, and ii) what are the different realizations of each point of the so called "variants".

In addition, variability occurs in the various phase of the software lifecycle [12]. This representation can cover the different phases of development of the system, from design to realization [13]. Consequently, it enables us to adapt and reuse the structure, the behavior of the software or the underlying processes. Generally, there are six steps in the variability management defined by [2]:

- The definition of variation points in the basis objects,
- The definition of the elements that can be potentially linked to these variation points,
- The definition of the relationships between these elements and variation points,
- The definition constraints between potential point's variations that may affect the instantiation, i.e. it shows the choices in the variability model.
- The specification of knowledge expressed in the variability,
- The specification of the different variations resolution mechanism.

Variability is all about choices that enable engineers to extend, change, customize, or configure product family artifacts to be used in a specific context. These choices emerge during all phases of the derivation process, such as requirements elicitation, the configuration of the code, and testing. In our classification framework, the concept of choice refers both to how the ability to choose is modeled, as well as how the possible options that can be selected are modeled [11].

2.2. Variability in SOA

SOA is an architectural style based on services. Firstly, all services are able to satisfy the needs of a user. Consequently, SOA has become more and more popular as a way to support the fast development of new applications. The major properties of a SOA solution are loose coupling, the independence of the technological aspects and scalability. The loose coupling property implies that a service does not directly call another service.

Therefore, it is easier to re-use a service because it is not directly linked to another.

The reusability of individual services is determined by their ability to support the variability required to adjust them to different contexts. This means, if services or service-based systems cannot adapt to changing situations and contexts, they can only be reused in a very limited scope [10]. Therefore, enhancing variability in service-based systems and providing methods that help explicitly model and manage variability facilitates highly reusable and configurable services and adaptable to specific contexts. In particular, the main problem of the variability in the transactional web service composition becomes visible in the response to the preferences of different users on demand services in different situations, which requires different actions to take [14].

Moreover, users' interventions expect a higher level of abstraction in the process of service creation. They are unable to use high technical and complex tools [15]. A number of approaches identify a variable mechanism to change the structure of the web services composition on the basis of each user intervention, integrating information of the variability in each model, in order to model variability at the architectural level [14]. At the level of the service interface, variability occurs when replacing a service by another with different interfaces, or to the evolution of the interface of a service by its supplier, while respecting the state of syntactic and semantic conflicts on the input and output messages.

3. The Proposed COMPARATIVE Framework

The proposed comparative framework considers the variability according to four different views. As illustrated in Fig. 1, each view allows capturing a relevant aspect in the management of variability.

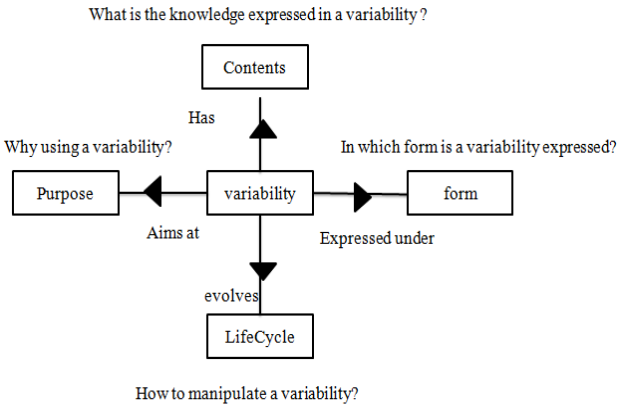

Figure 1. The four views of variability.

The four views are as following:

- The form describes the variability in static, animated and interactive form.
- The content concerns the type of knowledge, structure and cover that are expressed in the variability. Variability focuses on the management and modeling of system functionality to describe a wider flexibility in companies.
- The purpose is used to capture the role of variability. Describing the functionality of a system, explaining drawbacks or inefficiencies of a system are examples of roles that can be assigned to variability.
- The lifecycle suggests considering the variability as scalable object in time through the execution of operations.

Each view is characterized by a set of facets. Each facet has a metric measured by a set of pertinent attributes. Attribute values are defined within a domain. A domain may be a predefined type (Integer, Boolean, ...), an enumerated type (ENUM {x, y, z}), or a structured type (SET or TUPLE). So we describe in detail the suggested classification in the next section.

3.1. The Form View

In order to capture the variety of the different variability descriptions we introduce a facet named the

description. One or many formalisms can be used to describe the variability [16].

The structured text description is presented as an informal description, containing a list of conditions expressed with a semi formal language.

The tables are mainly used to highlight the variants with their relating choices. They are usually arranged with one row per variant and one column for every choice [17]. For instance, the GQM is one of the formalism that is used in this situation. Another example of table is used to present an explanatory table showing the different variants of invoice creation process in a billing product [19].

When the system contains different interrelated parts having different features, a diagram is used to describe this relation. Two types of diagram can be used : structural and behavioral. For example, COVAMOF in have designed a UML profile to model the variability at the architectural level of web-based systems, by exhibiting several types of UML diagram.

Furthermore, knowledge representation formalism might be used to describe the knowledge related to the variants. For instance, ontologies are used as formalism for the representation of knowledge related to features models in service-compositions as shown in Figure 2.

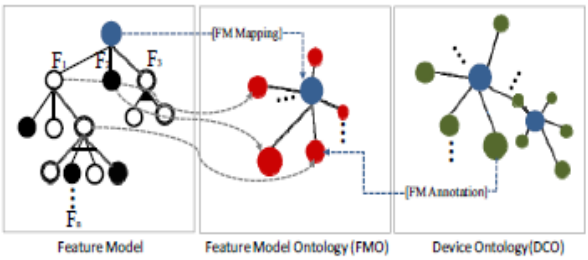

Fig. 2. Feature model mapping and annotation by NFRs ontology

To sum up, variability may be represented in a variety of form, either structured text, table, diagram, or knowledge representation formalism. Classifying a variability-based approach along the description facet comes down to give its values to the following attributes:

Form: SET (ENUM {structured text, table, diagram, knowledge representation formalism})

Formalism name: string

3.2. The Contents View

The content may vary from behavioral information (actions, events, activities) to objects (entities, data, attributes...). Other typical contents are organizational information (structure of company, groups, departments, agents, etc.) and stakeholder information i.e. knowledge (characteristics of people, their views and aspirations and wishes, aims and objectives).

In order to capture the variety of aspects related to the contents view, the framework introduces three facets, the structure facet, the knowledge facet and the coverage facet.

3.2.1. The structure facet

This facet aims to comparing the approaches according to the management and modeling of the amount of variability information they capture, including organizational information, social settings, goals, etc. therefore, we make the difference between the internal behavior of the system, the system interactions with its environment, and the organizational context of the system.

The authors in [20] propose a framework (internal system) that provides models, tools and objects to support the dynamic adaptation of service compositions. They introduce mechanisms to express where and how service compositions can be adapted. The SaaS applications (internal system) in [14] are built according to the SOA model (Service Oriented Architecture). This choice is related to the industrial needs in terms of standardized interactions between distributed and heterogeneous systems (system interactions with its environment); this can be easily achieved through the use of web services (SOA implementation technology). Several approaches, such as [1,2,6,27,34], present the interaction of various aspects by using the diagrams, Feature model, etc. In

addition, Galster et al., [5] seek to improve interaction between governments (internal system) and citizens.

The organizational context can be considered part of the definition of the application domain. Paper [5] studies the constraints for services oriented architectures of intensive variability. The constraints are understood as particular types of architectural preoccupations (ie, the interests of stakeholders in architecture) and interpreted as problematic elements imposed by the environment on the reference architectures design process.

In summary, we have defined four types of structural information that might be included in the content. This leads to the following attributes to characterize the variability-based approaches:

- System internal: Boolean
- System interaction: Boolean
- Organizational context: Boolean

3.2.2. Knowledge facet

The knowledge facet includes specific knowledge1† related to variability regarding any aspect of SOA. The knowledge implies the ability to mobilize information to act. It may contain rules, facts or any other representation used in the decision-making. In SOA, the knowledge can be related to the technology aspects, the environmental aspect, the platforms aspect or the security aspect, etc.

So we lean on the [9] to suggest a distinction between these aspects by defining the following attributes:

- Descriptions of alternative solutions to a problem (Positions): Boolean
- Arguments for objecting or supporting a given position (Arguments) : Boolean
- Descriptions of problems or conflicts (Issues): Boolean
- Choices of a particular position (Decisions): Boolean.

In this paper, we are interested particularly to the dimension of decision. The decisions making may be conducted automatically based on constraints and earlier decisions. The decisions making that are taken should lead to a real choice or product. For example, Alférez et al. in [20] present a variability model that expresses decisions leading to different variations at runtime.

The paper [23] is one of the most significant in the knowledge expression. The authors discuss different variability models in SOC solutions (argument). They compare these models with each other against the variability evaluation factors. This model covers problems and solutions (position), common and variable. Thus, they identify that certain models can also be combined together for a better solution or to solve variability problems (decision).

Finally, the two approaches [24] [29] lack knowledge and indicate only the position.

3.2.3. Coverage facet

The coverage facet aims to classify variability-based approaches according to the kind of information they capture. Our classification distinguishes between functional, non-functional and intentional aspects.

The functional attribute

The functional aspects can be captured through three design artifacts: structure, function and behavior [25][9]. The majority of the variants models focus on the descriptions of behavioral aspects.

The Fig. 3, adapted from [22], contains structured descriptions illustrating a variability defined at the functional level.

> *In order to ensure development and reuse services supporting the variability, a framework called SV3R have been proposed [22]. This framework can be understood according to three components:*
>
> ***Configurable services producer:*** *This component represents the responsible for the production of variable services called in this approach configurable services.*
>
> *....*
>
> ***Configurable Service Registry:*** *This component is used to save the various*

[1]† Knowledge: http://www.businessdictionary.com/definition/knowledge.html

components of configurable services

....

Configurable Service Customer: *This component represents the application developer by the reused services.*

Fig. 3. Adapted extract by [22].

Most of the variability approaches [5], [20], [24], [29] consider this functional attribute .

The intentional attribute

We suggest refining the intentional attribute further using the following set of concepts: Goal, Problem, Responsibility, Opportunity, Cause, and Goal dependency. In [5][26][27][28], the authors indicate how high level constraints on services, which specifies objectives, can be used to express an intensive variability design. For instance, in [14] the authors used a high level constraints in the variability description and, in order to meet the varying requirements of each tenant, they provided a customizable and configurable web services composition at runtime.

The non-functional attribute

The non-functional attribute defines the variation related to the quality constraints, namely the performance, time constraints, cost constraints, user support, backup / recovery, maintainability, flexibility, portability, security / safety, design constraints, etc. In [5], one of the constraints presented by the authors is the quality attribute. For example, in an e-government system and from the view of a municipality, a reference service-based architecture must be able to work in a large scale (eg, local e-government, central government), to integrate many external systems and to ensure high reliability and low downtime.

To sum up, the three attributes of the coverage facet are associated with the following values:

- Functional: SET (ENUM {Structure, Function, Behaviour})
- Intentional: SET (ENUM {Goal, Problem, Responsibility, Opportunity, Cause, Goal dependency})
- Non Functional: SET (ENUM {Performance, Time constraints, Cost constraints, User support, Documentation, Examples, Backup / Recovery, Maintainability, Flexibility, Portability, Security / Safety, Design constraints, Error handling })

3.3. The Purpose View

Generally, there exist a certain purposes to use variability. The purpose attribute allows variability to be classified according to the role they aim to play in web service.

3.3.1. The descriptive variability

In descriptive variability, the analyst and the user are mainly involved to capture requirements. Reusability of individual services and composition of services are determined by taking into consideration the different contexts [17] [18]. In [20] and [31] authors provided a formalism to define the context and introduced mechanisms to adapt the service composition at runtime. Also, they have provided an infrastructure that detects changes in context.

3.3.2. The exploratory variability

Exploratory variability serve when several different possible solutions satisfy a given set of requirements [32]. The different choices cannot be considered at design time and have to be explored and evaluated at runtime.

The exploratory variability demonstrated in [29], by adding warehouses variation point in the SCMS architecture to select one of two warehouses, is one of the illustrative example. In [21], [27] and [35], the authors found that neither SoaML nor UML has native support for exploratory variability modeling, they used a meta-model function based on UML. The variability meta-model identifies how different meta-classes relate to each other and in connection with this they showed how to implement an exploratory variability.

3.3.3. Explanatory variability

Explanatory variability is useful in situations that raise issues and require explanations about the rationale of these issues [28]. For example the approach [5] includes the variability explanations and reasons by questions and imposed constraints.

In summary, a variability-based approach will be classified according to the role it aims to play by three Boolean values associated to the three following attributes:

- Descriptive: Boolean
- Exploratory: Boolean
- Explanatory: Boolean

3.4. The Lifecycle View

How variabilities are captured, evolved and improved is the concern of the lifecycle view. The lifecycle view helps classifying variability based approaches depending on the way they handle these dynamic aspects.

3.4.1. The lifespan facet

Variability have a lifespan. The framework distinguishes between transient variability used as temporary items from persistent variability which persist over time.

It is worth mentioning that most of the existing approaches are based on the persistent variability.

In summary, the variability aspects are classified according to two values, transient and persistent:

Lifespan: ENUM {transient, persistent}

3.4.2. Lifecycle facet

This facet is concerned with how the variability are captured, evolved and are eventually transformed during its definition. The lifecycle facet aims to classify the variability according to the types of operations they carried out.

Operations on variability can be classified in the following types: Capture operations, Refinement operations, Integration operations, Expansion operations and Delete operations.

Most of the existing approaches hardly suggest any alternative to the creation of variability from scratch. Meanwhile, variability is captured through reusing has already been proposed.

To sum up, the attribute of the lifecycle facet is associated with the following values

Lifecycle: SET(ENUM {capture, Refinement, Integration, Expansion, Delete})

4. Review of Variability Based Approaches According to the Framework

First of all, we present the most known approaches managing the variability. Thereafter, we compare these approaches according to the proposed comparison framework.

The main variability-based approaches in SOA

4.1. Galster *et al.*

Galster et al. [5] put forward an approach that analyzes the variability written in natural language and structured by constraints imposed on the reference architecture. The variability can be seen through their textual and architectural representation that organizes the list of constraints.

The approach comprises the behavior modeling of an internal system, system interaction and context organization. In the knowledge perspective, authors consider almost all attributes, but they do not indicate decisions on choice of constraints by reference architectures of intensive variability. It is clear that the internal behavior and interactions are mainly addressed at the functional level. Thus, all functional attributes are considered. The Intentional attributes are the most studied in this paper. It treat the purpose, responsibility and opportunity. Otherwise, they focus on non-functional properties, indicating a constraint to the quality attributes. The quality attributes required by municipalities include performance and scalability. From the perspective of a municipality, a reference architecture based on service must be able to integrate into a large scale. In particular, it ensures high reliability and low downtime. The confidentiality and security issues reduce the desire to use cloud computing in the context of SOA.

In lifecycle view, the approach proposes transient variability because of changing situations. No alternative generates variability from scratch. Other operations support the following two activities: (i) integration is also

supported in the reference architectures. (ii) Refining screw transform variations to make them easy to understand or more reusable. The expansion and suppression are not indicated.

4.2. Ghaddar *et al.*

Ghaddar et al., [14] write an approach based on the variability managing. They specify and elaborate the variability in several architectural layers in SaaS applications. The variability is described in natural language and is related to the architecture..

The variability defined in the three architectural layers, which implies solutions diversity provoking the isolation and a lack of consistency. This is why the authors propose a comprehensive and unique model of the variability of the application as internal system. For this the authors propose a global and unique model of the application variability as internal system. The Variability is defined as a behavioral specification; it captures how the system reacts with its environment. This definition also provides information about the organizational context for the knowledge attributes (position, argument).

The authors focus in the intentional aspect of variability, neglecting the functional and structural interest.

The approach identified a descriptive work that explains the work context and the gaps to meet the specific needs of a tenant without affecting other hosted tenants. Thus, this work explains the need for new approach, which will reduce the lack of uniform model for the management and modeling of the variability within the SaaS architecture. Exploratory variability reflects a variability management methodology representing the variations of the multi-tenant application as a model of abstract and uniform variability. This can be achieved by exploiting the well-known modeling techniques, after by choosing the techniques and the most appropriate application layer to implement the variations.

4.3. Alférez *et al*.

Alferez et al., displays different forms to represent the variability that can be taken. The examples use a variety of form (text, graphics, architecture, table, and graph).

In the knowledge aspect, the variability model has variation points that express the decisions leading to different variations of execution. only the questions were absent in this paper.

The authors discuss the functional attributes pointing to the three aspects related to this attribute. So, in the approach [20], few intentions can be captured. The non-functional attribute addressed in this approach, indicates the error situations, performance and quality of service.

Finally, the set of variabilities in the dynamic adaptation model of service compositions are varied. The variability lifecycle is transient, which can evolve with their relationships with design specifications. Only integration and refining are considered in the lifecycle facet.

4.4. Chakir and Fredj

The approach [22] suggests a new framework for the service variability managing. This model describes using several form such as text, graphics and architecture. The internal system proposed by [22] is called SV3R. Thus, it is based on the interaction between the models component to treat the changes. The knowledge aspect has two dimensions, position and argument.

The SV3R descriptions contain all intentional, functional requirements, and the aspects quality. The services appearance has lively aroused interest in introducing variability in the first stages of the services development and elaboration. Indeed, the variability concept finds its origins in the product lines engineering. Latter has objective to minimize the cost of software development in a particular field of application, and improve reuse.

Finally, the framework goal is to capture the variability at different abstraction levels. Hence, it aims to transform variations to make them easy to understand or more reusable.

4.5. Koning *et al*.

Koning et al., [24] provide a traditional and formal approach to the modeling and analysis of variability. Then

the variabilities are described using natural language with several forms.

The approach [24] does not include the internal system. Nonetheless, it organized the context to introduce concepts related to the variability of BPEL. The Knowledge structures are not included in this work.

The variability coverage is a behavior. Consequently, several types of variability need to be detected in the service composition to model in web services. The description and the exploration of VxBPEL expansion introduce new activities allowing an information variability modeling, especially the variation points, variants and realization relationships (which can improve the workability of the system variability).

The variability appear following functions to model. Thus, all functional attributes are considered. The Intentional attributes are addressed. Consequently, koning et al., Treat the purpose, responsibility and opportunity in this approach, but they do not focus on the non-functional properties.

4.6. Sun *et al.*

The variability in can be presented in the diagrams analyzed by the narrative texts. The approach of Sun et al., is more powerful using diagrams to model variability in the COVAMOF. The using case of variation point can capture the organization and interaction information of the system with the environment. The model Sun et al., [29] is similar to the model Koning et al., [24]. It covers all the functional and structural aspects indicating the behavior of the system, the objective and responsibility to organize work. Eventually, the exploratory variabilities are mentioned by adding variation point warehouse to the SCMS architecture to select one of the two warehouses.

Table 1. This is a Table

	Form	Content										Purpose			Lifecycle			
	Description	Structure			Knowledge				Coverage			D	E	E				
	Form	Sys internal	Sys interaction	Org context	position	Argument	Issue	Decision	Functional	Intentional	Non-functional	Description	Exploratory	Explanatory	Integration	Refinement	Expansion	Delete
Galser et al., [5]	ST, TA	✓	✓	✓	✓	✓	✓		S,F, B	R, O	SU,C,P, S, QoS	✓		✓	✓	✓		
Ghaddar et al., [14]	ST	✓	✓	✓	✓	✓			B	G, O, R	E,F	✓	✓	✓	✓	✓		
Alférez et al., [20]	ST, TA, D, KR	✓	✓	✓	✓	✓		✓	S,F, B	G, O, R	S,P,E, QoS	✓			✓	✓		
Abu-Mata r et al., [21]	ST		✓	✓	✓	✓			S,F, B	G, O, R	SU,P,F	✓	✓		✓	✓		
Chakir and Fredj, [22]	ST, KR	✓	✓	✓	✓	✓			S,F, B	G, O, R	SU,P,F, QoS	✓	✓		✓	✓		
Khan et al., [23]	ST, TA		✓	✓	✓	✓		✓	B	G, O, R	SU,P,F	✓	✓		✓	✓	✓	✓
Koning et al., [24]	ST, KR		✓	✓	✓				S,F, B	G, O, R	{}	✓	✓		✓	✓		
Sun et al., [29]	ST, D, KR		✓	✓	✓	✓			S,F, B	G, O, R	{}	✓	✓			✓		
Mohabbat i et al.,	ST, D	✓	✓	✓	✓	✓			S,F, B	G, O	S,C,P	✓		✓	✓	✓		

[33]																		

Boolean = ✓: True, whitespace: False
Form= ST: Structured Text, TA: Table, D: diagram, KR: knowledge Representation Formalism
Functional= S: Structure, F: Function, B: Behavior
Intentional= G: Goal, R: Responsibilities, O: Opportunity
Non-functional= {SU: User Support, S: Security, P: performance, C: time / QoS constraints, F: flexibility, E: Error}

4.7. Mohabbati *et al.*

In [33], the variability are presented in text form with some notations as the diagram and architecture. The modeled variability descriptions semantically use the ontology. This approach treats the service requirements specification that influences through the requests specification. The internal system behavior can capture information about the organization and system interaction.

The ontology is defined in this approach as a formal specification using knowledge representation contained in the models features. Feature Model Ontology (FMO) provides the semantic representation, relationships and dependencies and the functional constraints that express another form of relationship between the model features.

The system to [33] covers the functional and structural requirements aspects. So, the semantic modeling of variability requires the constraints imposed by the device system to attain the best level of security and system performance.

In addition, several types, namely summaries, work situation description, user preference variability and the variability explanation, present the variability description. Then the system configures the variability depending on the preferences and device capability.

The variability presented by [33], elaborated by refining operations that transform variants to make them easy to understand or more reusable.

5. Discussion

In this section, we discuss the approaches according to the four framework viewpoints and based on the metric of each facet. This synthesis allows clarifying how the variability is considered.

First of all, based on the table we notice that the definition of variability vary from one approach to another. Most of the approaches use a structured text to describe the variability. The more the description is based on a variety of formalisms, the more it is possible to manage different aspects of the variability.

When, the variability refers to the situations or behaviors, they focus on knowledge. In this context, we stressed the importance of the content view to distinguish between these approaches. This distinction is highlighted in different ways:

The Structure facet aims to classify the approaches depending on the management and the modeling of the information they capture, including organizational information, social settings, objectives, etc.

The knowledge facet includes the specific knowledge related to the variability aspect in web service composition. It may contain rules, facts or other representations before making the decision. The decisions that are taken, should lead to a choice or an actual product.

The cover facet classifies approaches depending on the types of information they capture.

So, we note the low concentration of the approaches considering the knowledge aspect, no approach treats all attributes in this facet. The attributes of the structural facet appear only in the studies [5], [14] [20] [22].

Furthermore, categorization in coverage facet focuses on the variability according to the users perception. The distinction between the approaches is quite clear in the non-functional attribute, because each approach shows its own value. The example of (Galster et al., 2013) [17] showed a significant number of metrics in the non-functional attribute, with the absence of the metric in (Koning et al., 2009) [24]and (Sun et al., 2010).[4]

Finally, the main purpose found in most approaches is to ensure flexibility in the SOA. Only few approaches provide better satisfaction of system requirements [14], [21]-[24] [29]. In addition, only very few approaches

[33]-[36] include the explanations and reasons for variability.

6. Conclusion

Recently, the variability appears in SOA to ensure high level of flexibility by quickly accommodating the system to the changing needs.

In order to understand the flexibility level in the existing approaches we developed a multidimensional framework. The different dimensions are used to identify and clarify the aspect of variability in the web service composition. The comparative framework, on the one hand, improves understanding and comparison of existing approaches by providing pertinent information on variability. On the other hand, the framework helps researchers to propose more innovative approaches.

Indeed, we studied the most known approaches and the discussion revealed the advantages and the limitations of such approaches.

Finally, the enhancement of knowledge integration in the management of variability will extend this work.

References

[1] Papazoglou, M. P., Traverso, P., Dustdar, S., & Leymann, F. (2007). Service-oriented computing: State of the art and research challenges. *Computer*, *(11)*, 38-45.

[2] Sbai, H., Fredj, M., & Chakir, B. (2015). Generating sevices supporting variability from configurable process models. *Journal of Theoretical and Applied Information Technology*, *72(1)*.

[3] Dinkelaker, T., Monperrus, M., & Mezini, M. (2010). Supporting variability with late semantic adaptations of domain-specific modeling languages.

[4] Sun, C. A., Rossing, R., Sinnema, M., Bulanov, P., & Aiello, M. (2010). Modeling and managing the variability of Web service-based systems. *Journal of Systems and Software*, *83(3)*, 502-516.

[5] Galster, M., Avgeriou, P., & Tofan, D. (2013). Constraints for the design of variability-intensive service-oriented reference architectures–An industrial case study. *Information and Software Technology*, *55(2)*, 428-441.

[6] Rusli, H. M., Puteh, M., Ibrahim, S., & Tabatabaei, S. G. H. (2011, May). A comparative evaluation of state-of-the-art web service composition testing approaches. *Proceedings of the 6th International Workshop on Automation of Software Test*.

[7] Czarnecki, K., Grünbacher, P., Rabiser, R., Schmid, K., & Wąsowski, A. (2012, January). Cool features and tough decisions: a comparison of variability modeling approaches. *Proceedings of the Sixth International Workshop on Variability Modeling of Software-Intensive Systems*.

[8] Mahdavi-Hezavehi, S., Galster, M., & Avgeriou, P. (2013). Variability in quality attributes of service-based software systems: A systematic literature review. *Information and Software Technology*, *55(2)*, 320-343

[9] Chen, L., Ali Babar, M., & Ali, N. (2009, August). Variability management in software product lines: a systematic review. *Proceedings of the 13th International Software Product Line Conference*.

[10] Kazhamiakin, R., Benbernou, S., Baresi, L., Plebani, P., Uhlig, M., & Barais, O. (2010). Adaptation of service-based systems. *Service Research Challenges and Solutions for the Future Internet*.

[11] Schmid K., & John, I. (August 2004). A customizable approach to full lifecycle variability management. *Science of Computer Programming*, *53(3)*, 259-284.

[12] Galster, M., Weyns, D., Tofan, D., Michalik, B., & Avgeriou, P. (2014). Variability in software systems-a systematic literature review. *IEEE Transactions on Software Engineering*, *40(3)*, 282-306.

[13] Chen, L., Ali Babar, M., & Ali, N. (2009, August). Variability management in software product lines: a systematic review. *Proceedings of the 13th International Software Product Line Conference*.

[14] Ghaddar, A., Tamzalit, D., & Assaf, A. (2012). Gestion de la variabilité dans les applications SaaS multi-locataire.

[15] Goncalves da Silva, E. M., Ferreira Pires, L., & Van, S. M. J. (2009). Supporting dynamic service composition at

runtime based on end-user requirements. *CEUR Workshop Proceedings*.

[16] Carbonnel, J., Huchard, M., & Gutierrez, A. (2014). Représentation de la variabilité par des treillis de concepts.

[17] Galster, M., & Avgeriou, P. (2013). Variability in Web Services. *Systems and Software Variability Management* (pp. 269-278).

[18] Loesch, F., & Ploedereder, E. (2007, March). Restructuring variability in software product lines using concept analysis of product configurations. *Proceedings of the 11th European Conference on Software Maintenance and Reengineering*.

[19] Jamoussi Y. (2015). Enhancing satisfaction of Actors' requirements in Web service composition: A Guided negotiation based Approach. *Journal of Software Engineering*.

[20] Alférez, G. H., Pelechano, V., Mazo, R., Salinesi, C., & Diaz, D. (2014). Dynamic adaptation of service compositions with variability models. *Journal of Systems and Software*, *91*, 24-47

[21] Abu-Matar, M., Gomaa, H., Kim, M., & Elkhodary, A. M. (2010, July). Feature modeling for service variability management in service-oriented architectures.

[22] Chakir, B., & Fredj, M. (2014). SV3R: Un framework pour la gestion de la variabilité de services.

[23] Khan, A., Kästner, C., Köppen, V., & Saake, G. (2011). Service variability patterns in SOC. *Tech. Rep, 5. School of Computer Science*, University of Magdeburg, Magdeburg, Germany.

[24] Koning, M., Sun, C. A., Sinnema, M., & Avgeriou, P. (2009). VxBPEL: Supporting variability for Web services in BPEL. *Information and Software Technology*, *51(2)*, 258-269.

[25] Lemrabet, Y., Benkeltoum, N., Bigand, M., Clin, D., & Bourey, J. P. (2011). Entreprise agile et l'alignement métier-IT avec les approches BPM et SOA: Retour d'expérience. Actes de la CIGI.

[26] Santos, E., Castro, J., Sanchez, J., & Pastor, O. (2010, April). A goal-oriented approach for variability in BPMN.

[27] Valença, G., Alves, C., Alves, V., & Niu, N. (2013). A systematic mapping study on business process variability. *International Journal of Computer Science & Information Technology*, *5(1)*.

[28] Van Der Aalst, W. M., Ter Hofstede, A. H., & Weske, M. (2003). Business process management: A survey. In Business process management (pp. 1-12). Springer Berlin Heidelberg.

[29] Patel, S. (2014, January). Practical problems with modeling variability in test cases: an industrial perspective. *Proceedings of the Eighth International Workshop on Variability Modelling of Software-Intensive Systems* .

[30] Sinnema, M., & Deelstra, S. (2007). Classifying variability modeling techniques. *Information and Software Technology*, *49(7)*, 717-739.

[31] Chang, S. H., & Kim, S. D. (2007, September). A variability modeling method for adaptable services in service-oriented computing. *Proceedings of the 11th International software Product Line Conference*.

[32] Rolland, C., & Nurcan, S. (2010, January). Business process lines to deal with the variability. *Proceedings of the 2010 43rd Hawaii International Conference on In System Sciences*.

[33] Mohabbati, B., Kaviani, N., & Gašević, D. (2009, August). Semantic variability modeling for multi-staged service composition. *Proceedings of the 13th Software Product Lines Conference* .

[34] Hallerbach, A., Bauer, T., & Reichert, M. (2010). Capturing variability in business process models: the Provop approach. *Journal of Software Maintenance and Evolution: Research and Practice*, *22(6-7)*, 519-546.

[35] Chakir, B., & Fredj, M. (2010). Towards a model driven approach for variability management in SOA. *Proceedings of the International Conference on Models of Information and Communication Systems*.

[36] Asadi, M., Mohabbati, B., Kaviani, N., Gasevic, D., Boskovic, M., & Hatala, M. (2009). Model-driven development of families of service-oriented architectures. *First International Workshop on Feature-Oriented Software Development*.

[37] Berger, T., Pfeiffer, R. H., Tartler, R., Dienst, S., Czarnecki, K., Wąsowski, A., & She, S. (2014). Variability mechanisms in software ecosystems. *Information and Software Technology*, *56(11)*, 1520-1535.

Yassine Jamoussi (HDR between University of Manouba, Tunisia – University of Toulouse, France) is an associate professor at the National School of Computer Science, Tunis Tunisia and he is currently a Faculty member at the department of Computer Science at Sultan Qaboos University. His research interests include process-centered environment, modeling and meta-modeling of flexible process, process enactment, modeling variability, adaptation, evaluation of process model, service oriented architecture. His research work has been supported by several funding, such as CMCU (Europeen), INRIA (French), MRT (Ministry of Research and Technology and Industry of Tunisia), IG (SQU Oman). He has been invited to present his research in many countries in North America, Europe, Africa and in the Middle East. Dr Yassine Jamoussi has been conference chair of IEEE conferences and member of over 20 program committees. He published 20 Journal and 40 conference papers in International journals and conference proceedings.

Sabrine Bahi is currently pursuing PhD degree in computer science at National School of Computer Science of the Manouba University (Tunisia). She is member of RIADI Laboratory and her interests include software engineering and variability in SOA

.