A Semantic-Aware Framework for Composite Services Engineering Based on Semantic Similarity and Concept Lattices

Ahmed Abid[1,3](✉), Nizar Messai[1], Mohsen Rouached[2], Thomas Devogele[1], and Mohamed Abid[3]

[1] LI, University Francois Rabelais Tours, Tours, France
ahmed.abid@etu.univ-tours.fr,
{nizar.messai,thomas.devogele}@univ-tours.fr
[2] College of Computers and Information Technology, Taif University, Taif, Saudi Arabia
m.rouached@tu.edu.sa
[3] CES Laboratory, Sfax University, Sfax, Tunisia
mohamed.abid@enis.rnu.tn

Abstract. This paper presents a semantic framework called IDECSE for composite Web services modeling and engineering. This framework uses semantic similarity measures and Formal Concept Analysis formalism to generate classes of similar services that can be composable to satisfy users queries and preferences. A reasoning mechanism is also proposed to produce reliable composite services. By considering semantics for describing, discovering, composing, and monitoring services, IDECSE addresses the challenge of achieving a full governance of the composition process.

Keywords: Semantic web services · Semantic similarity · Formal concept analysis · Web services composition

1 Introduction

Today, business processes are increasingly implemented by dynamically composing Web services seen as the main contribution the Service Oriented Architecture (SOA) brings to enterprise business process automation. Therefore, Web services composition has became an attractive way of developing value added Web services and would lead to substantial gains in productivity in several application domains including e-Enterprise, e-Business, e-Government, and e-Science. However, an important problem with current services compositions concerns their life-cycle and their management, also called their governance. The challenge is how to achieve a full governance of the composition allowing the continuous and dynamic improvement of the composition to support and encourage the adoption of SOA technologies. IDECSE [1] addressed the above challenge and proposed an integrated declarative framework to bridge the gap between the

S. Nurcan and E. Pimenidis (Eds.): CAiSE Forum 2014, LNBIP 204, pp. 148–164, 2015.
DOI: 10.1007/978-3-319-19270-3_10

process modeling, verification and monitoring and thus allowing for self-healing Web services compositions. Another challenge concerns the semantic gaps in the definition of atomic Web services provided by autonomous and different services providers, and therefore composition modeling frameworks should provide support for bridging such semantic gaps. Supporting environment dynamicity is also a critical requirement since available services as well as user requirements change frequently over time. Thus, the environment for service engineering needs to support rapid and dynamic re-design through appropriate and automatic tools for dealing jointly with adaptation at modeling, deployment and run-time.

While numerous composition approaches have been developed, very little has been done towards dealing with these challenges because of their complexity. Consequently, existing frameworks need to be enhanced using new semantic-aware methods and tools. In this paper, we propose to address this issue by enhancing IDECSE approach with semantics in all composition steps. Mainly, improvement of IDECSE consists in considering service similarity measures.

The rest of the paper is organized as follows. Section 2 presents the architecture of the improved IDECSE framework and details its modules. We mainly focus on the Service Classification Module. In Sect. 3, we discuss the current implementation and the ideas to validate the proposed approach. Section 4 exposes literature review. Finally, Sect. 5 concludes the paper and outlines future work.

2 IDECSE Framework

A major problem with current services compositions concerns their life-cycle and their management, also called their governance. The challenge is how to achieve a full governance of the composition allowing the continuous and dynamic improvement of the composition to support and encourage the adoption of SOA technologies. Many SOA management initiatives fail to get off the drawing board once systems architects recognize the scale of integration work to bring the different elements of functionality into play. IDECSE [2] addressed this challenge by proposing an integrated framework to bridge the gap between the process modeling, verification and monitoring and thus allowing for self-healing Web services compositions. This framework aims also to fully integrate semantics in all stages of the composition global life-cycle. First, user requirements are better understood using refinement techniques such as generalization or specification of concepts from a given ontology. IDECSE appeals for data mining techniques for classifying and mining services into Service Registry based on semantic relations. The main components of the IDECSE architecture are depicted in Fig. 1. It consists of five modules covering the global composition life-cycle (i.e. specification, modeling, composition, deployment, and monitoring). These modules are described and detailed in the following sections.

2.1 Service Request Module

The Service Request module (First layer in Fig. 1) translates the user requirements to an internal language to be used by the Service Classification module and

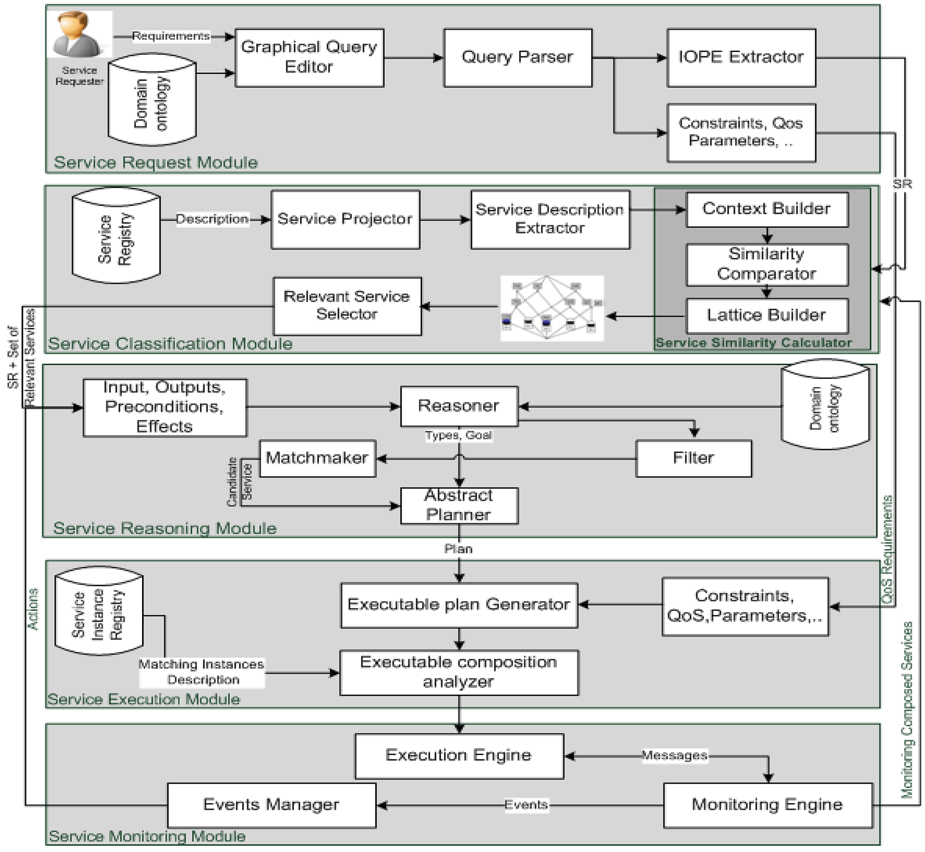

Fig. 1. IDECSE architecture

the Service Reasoning module. The Graphical Query Editor relies on a domain ontology to analyze user requirements before enriching them through adding new ontology concepts based on semantic relations such as generalization, specialization, etc. The query is then parsed to extract functional and non-functional requirements. Functional requirements are modeled using the IOPE Extractor, which extracts the Inputs, Outputs, Preconditions and Effects.

Extracted requirements are then modeled as a new requested service called S_R. Given a domain ontology O, a user query Q, modeled as S_R, consists of a set of provided inputs $S_{R_{in}} \subseteq O$, a set of desired outputs $S_{R_{out}} \subseteq O$, a set of preconditions $S_{R_{pre}} \subseteq O$, a set of effects $S_{R_{eff}} \subseteq O$, and a set of quality of service constraints $S_{R_{qos}} = \{(q_1, v_1, w_1), (q_2, v_2, w_2), ..., (q_k, v_k, w_k)\}$, where $q_{i(i=1,2,...,k)}$ is a quality criterion, v_i is the required value for criterion q_i, and w_i is the weight assigned to this criterion such that $\sum_{i=1}^{k} w_i = 1$, and k the number of quality criteria involved in the query. We can model S_R as $S_R = \sum IOPE + \sum QoS$.

2.2 Service Classification Module

To deal with the important number of Web services and instead of considering the whole Service Registry, this module allows to classify available services semantically into classes according to their similarities. Its second role is to return only relevant services to S_R from the registry. This module contains four main components which are: Service Projector, Service Description Extractor, Service Similarity Calculator, and Relevant Service Selector.

To introduce the functionality of the proposed module for Web service classification, an example of semantic Web services is presented in Table 1.

Table 1. A set of services with their operations

Services	Id	Operations	Id
AUTHOR FINDER	WS_1	Find Author(B)	op_{11}
AUTHOR PRICE FINDER	WS_3	Find Author(B)	op_{21}
		Find Price(B)	op_{22}
SEARCH BOOK	WS_2	Find Book(A)	op_{31}
CURRENCY CONVERTOR	WS_4	Convert (C,C,P)	op_{41}
DOLLAR2EURO	WS_5	Dollar2euro(A)	op_{51}

The role of the Service Classification components is described and illustrated in the following parts.

1. **Service Projector:** The Service Projector selects services capabilities based on syntactical and semantic description for each service into one interface. Different semantic description languages were proposed, OWL-S and WSMO are the most important ones for this purpose. Both WSMO and OWL-S have the aim of providing the conceptual and technical means to realize Semantic Web services. The comparison done in [3] shows that although the aims of both WSMO and OWL-S are the same, they present some differences in the approach they take to achieve their goals especially in the definition of the process model and the grounding of Web services. OWL-S is more mature and is therefore considered in the IDECSE framework. The role of the Service Projector Module is to select service capabilities from OWL-S and WSDL descriptions and transmit them to the Service Description Extractor. In the running example, the operations of each service shown in Table 1 are the results transmitted by the Service Projector.
2. **Service Description Extractor:** Extracting comparator parameters from service capabilities is a mandatory step to measure similarity between services. The Service Classification Module relies on the functional properties of services which will be the parameters to extract from the OWL-S files.

   Semantic and Syntactic functional parameters of services are given in OWL-S description. In fact OWL-S is an ontology for services. Each service class in OWL-S refers to a declared semantic service [4]. Each service description is composed of three main parts: Service Profile, Service Model, and Service Grounding. In the following sections, we select relevant data to be compared from these parts:

a. Functionality Description in the Service Profile: The Service profile consists of four main parts. The first part describes the links between the Service profile and the service and its process model. The second part describes the contact information, intended for human consumption. The third part describes the functionality in terms of Input, Output, Precondition and Effect (IOPE). The last part describes the attributes of a profile. Relevant functional information about services exist in the last two parts. The Profile class defines the following properties for IOPE *hasInput* ranges over the Inputs, *hasOutput* ranges over the Output, *hasPrecondition* specifies a precondition of the service, *hasResult* specifies under which condition outputs are generated.

b. Functionality Description in the Service Model: The Service Model gives a detailed description on how to interact with the service. It can be used to supplement initial similarity measure by giving a more detailed perspective on the service internal workings.

c. Functionality Description in the Service Grounding: The Service Grounding specifies the details of how to access a service. It is not required in measuring service similarity because it can be the same especially for services from the same organization, but it provides a useful way of allowing users to specify the way of using the service.

In the syntactic side, IDECSE relies on the types of inputs and outputs of services. The Service Description Extractor tries to extract those parameters with their attributes and transmits them to the Service Similarity Calculator. Considering the previous example, Table 2 shows the different details of services. We note that the example uses simple services based on input and output functional parameters.

Table 2. Detailed Web services Description

Operations	Input concept	Output concept	Input Type	Output Type
op_{11}	Book	Person	String	String
op_{21}	Book	Person	String	String
op_{22}	Book	Price	String	Float
op_{31}	Text	Book	String	String
op_{41}	Price, Currency	Price	String/Float	Float
op_{51}	Price	Price	Float	Float

3. **The Service Similarity Calculator:** After selecting relevant parameters from services description, the Service Similarity Calculator measures the similarity and uses data mining techniques in order to classify available services

into classes according to their similarity. The main parts of this module and their functionalities are described below:

a. Context Builder: The Context Builder is responsible for preparing the input dataset to the classification module. It selects the main properties of Web services and creates a tabular representation where the rows correspond to the Web services, the columns correspond to the services capabilities (descriptions) such as type of input or the ontology that the input refers to, and finally table cells contain real values of these properties for each service.

b. Similarity Comparator: It can be considered as the main component of this module. The Similarity Comparator relies on a set of formulas and relations used to calculate similarity between services. Two types of formulas are taken into account by the IDECSE framework: the first is for the semantic and the second is for the syntactic Similarity and relatedness.

In the semantic field, and based on the selected parameters in the Service Description Extractor module, IDECSE defines a semantic measure function. Let S be a set of services ($|S| = n \in N$) and let σ be a similarity measure function $\sigma : S \times S \rightarrow [0\,, 1]$ which verify the conditions below:

– $\sigma(S_i, S_i) = 1 \forall\ i \in \{1, .., n\}$,
– $\sigma(S_i, S_j) = \sigma(S_j, S_i) \forall\ i \in \{1, .., n\}, \forall\ j \in \{1, .., n\}$.

An OWL-S service similarity can be defined as follows: For $i, j \in \{1, .., n\}$,

$$\sigma(S_i, S_j) = U_1 Sim_P(S_i, S_j) + U_2 Sim_M(S_i, S_j) + U_3 Sim_G(S_i, S_j) \quad (1)$$

where $\sum_{k=1}^{3} U_k = 1$

The functions Sim_P, Sim_M and Sim_G present respectively the similarity function between tow Services Profiles, Models and Groundings. As Service Model and Service Grounding are used to supplement the initial similarity and to specify details of how to access a service, their parameters can be modeled as Quality of services thus more importance is given to the functional similarity.

$$\sigma(S_i, S_j) \approx Sim_P(S_i, S_j) = W_1 Sim_I(S_i, S_j) + W_2 Sim_O(S_i, S_j) + W_3 \\ Sim_P(S_i, S_j) + W_4 Sim_E(S_i, S_j) \quad (2)$$

where $\sum_{k=1}^{4} W_k = 1$

Sim_I, Sim_O, Sim_P, Sim_E are respectively the similarity function between two services Input, Output, Precondition and Effects. Each parameter is semantically annotated with respect to an OWL concept. Thus, the functional similarity measurement between services is mapped to Ontology-based similarity. Studying ontological concepts, their details are divided into tow types which are: Type of Concept and Relation of Concept. Otherwise, measuring similarity between services in the IDECSE framework is based on semantic and syntactic similarity.

Regarding semantics, being machine interpretable and constructed by experts, ontologies present a very reliable and organized knowledge source system. For these reasons, ontologies have been extensively exploited in knowledge-based systems and, more precisely, to compute semantic similarity. Measures on the ontology based similarity are divided into three main categories which are: Edge-counting approaches [5,6], Features-based approaches [7] and Information Content approaches [8,9]. After surveying different Ontological similarity measures and their application situations and based on experimental results and benchmarks tests [10,11], the measure proposed in [12] gives pertinent results in our case. The formula is given below:

Let A and B be two concepts, represented by the nodes a and b in a predefined is-a semantic taxonomy (ontologies). Let C be a set of concepts of a given ontology, ($\leq$) is defined as a binary relation $\leq$: $C \times C$. For two concepts c_i and c_j, $c_i \leq c_j$ is fulfilled if c_i is a hierarchical specialization of c_j or if $c_i \equiv c_j$ (i.e. same concept). The set of taxonomical features describing the concept a is defined in terms of the relation $\leq$ in (3).

$$\phi(a) = \{c \in C | a \leq c\} \tag{3}$$

[12] defines the semantic similarity between tow concepts as follow:

$$Sim(a,b)_{sanchez} = 1 - \log(1 + \frac{|\phi(a) \backslash \phi(b)| + |\phi(b) \backslash \phi(a)|}{|\phi(a) \backslash \phi(b)| + |\phi(b) \backslash \phi(a)| + |\phi(a) \cap \phi(b)|}) \tag{4}$$

Compared to other measures based on taxonomical knowledge, the exploitation of the whole amount of unique and shared subsumers seems to give solid semantic evidences of semantic resemblance. Results show that the measure proposed by [12] surpasses basic Edge-Counting Features and Information content measures.

Moving to syntactical similarity, [13] proposed a practical measure between different data types, Table 3 groups different data types and Table 4 gives the similarity between different data types.

Table 3. Simple DataType groups [13]

Group	Simple Data Types
Integer Group	Integer, Byte, Short, Long
Real Group	Real, Float, Double, Decimal
String Group	String, NormalizedString
Date Group	Date, DateTime, Duration, Time
Boolean Group	Boolean

Table 4. Simple DataType groups similarity [13]

	Int.	Real	Str.	Date	Bol.
Int.	1.0	0.5	0.3	0.1	0.1
Real	1.0	1.0	0.1	0.0	0.1
Str.	0.7	0.7	1.0	0.8	0.3
Date	0.1	0.0	0.1	1.0	0.0
Bol.	0.1	0.0	0.1	0.0	0.1

Finally to compute semantic similarity between services, IDECSE framework combines the tow measures proposed in [12] and in [13]. The first one is used as a semantic measure based on the ontological features of concepts and the second is based on the syntactic measure between inputs and outputs types of services. Based on similarity measure given in (4), we redefine our similarity measure (2) in (5).

$$\begin{aligned}\sigma(S_i,S_j) = W_1[&N_{11}Sim_{I_{[12]}}(S_i,S_j) + N_{12}Sim_{I_{[13]}}(S_i,S_j)] + W_2\\ &[N_{21}Sim_{O_{[12]}}(S_i,S_j) + N_{22}Sim_{O_{[13]}}(S_i,S_j)] + W_3\\ &[N_{31}Sim_{Pre_{[12]}}(S_i,S_j) + N_{32}Sim_{Pre_{[13]}}(S_i,S_j)]+\\ &W_4[N_{41}Sim_{Eff_{[12]}}(S_i,S_j) + N_{42}Sim_{Eff_{[13]}}(S_i,S_j)] \quad (5)\end{aligned}$$

where $\sum_{k=1}^{4} W_k = 1$, $\sum_{j=1}^{2} N_{aj} = 1$ and $a \in \{1,2,3,4\}$.

The function σ is used then to calculate similarity between concepts that are referred by functional parameters of services. Finally, this module returns a Similarity Matrix (SimMat) containing similarity measures between available services. The Similarity Matrix generated from the proposed example using (5) is given in Table 5.

From $SimMat$, we can mainly extract several binary contexts using a threshold $\theta \in [0,1]$. Values of SimMat, which are greater or equal to the fixed threshold θ, are scaled to 1 and other values are scaled to 0. The binary context that corresponds to an arbitrary threshold for operations and services are shown in Tables 6 and 7. The SimCxt is a triple (O,O,R_{Sim_θ}), where O is a set of operations and R_{Sim_θ} is a binary relation indicating whether an operation is similar to another operation or not. The Context Matrix (Table 7 in our example) is the result transmitted to the next module to build the Lattice.

c. Lattice Builder: In this sub-module a Lattice of operations is built according to the Formal Concept Analysis (FCA)[14] formalism and its extension to complex data called Similarity-based Formal Concept Analysis (SFCA)[15]. An example of obtained Lattice for a binary context using the ConExp[1] tool is given in Fig. 2. This Lattice shows the grouping of similar operations.

[1] http://conexp.sourceforge.net/.

Table 5. The operations SimMat

	op_{11}	op_{21}	op_{22}	op_{31}	op_{41}	op_{51}
op_{11}	1	1	0.40	0.35	0	0
op_{21}	1	1	0.40	0.35	0	0
op_{22}	0.40	0.40	1	0.32	0	0
op_{31}	0.35	35	0.32	1	0	0
op_{41}	0	0	0	0	1	0.45
op_{51}	0	0	0	0	0.45	1

Table 6. The operation SimCxt for $\theta = 0.4$

	op_{11}	op_{21}	op_{22}	op_{31}	op_{41}	op_{51}
op_{11}	X	X	X			
op_{21}	X	X	X			
op_{21}	X	X	X			
op_{31}				X		
op_{41}					X	X
op_{51}					X	X

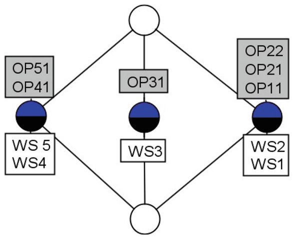

Fig. 2. Generated lattice for (SimCxt) shown in Table 7

4. **Relevant Service Selector:** Once the Lattice of services is built, the Relevant Service Selector identifies the most relevant classes of services from the Lattice. The Relevant Service Selector has two main roles. The first role is to select the most relevant services for the purpose of substitution process and thus to maintain a composite Web service application functionality as much as possible. The second role is about finding equivalent, similar and composable services to S_R requested by users in order to transmit them to the reasoning module.

Table 7. The service SimCxt for $\theta = 0.4$

	op_{11}	op_{21}	op_{22}	op_{31}	op_{41}	op_{51}
WS_1	X	X	X			
WS_2	X	X	X			
WS_3				X		
WS_4					X	X
WS_5					X	X

5. **Example of User Query:** We consider a simple user query Q = "Find an author of a Book based on its title" in order to demonstrate the usefulness of the Service Classification Module in the IDECSE framework.

Table 8. S_R detailed descriptions

Op	Input	Output	In. Type	Out. Type
op_R	Title	Author	String	String

Parsing the query is the role of the Service Request Module. Table 8 details S_R description generated from this module. Then Table 8 is transmitted to the Service Similarity Calculator sub-module. After adding S_R to the existing context, a similarity measure is then calculated between S_R and the rest of available services. We suppose here that available services are those given in Table 1. Results of similarity measures are shown in Table 9.

Table 9. S_R similarity measures

	op_{11}	op_{21}	op_{22}	op_{31}	op_{41}	op_{51}
op_R	0.7	0.7	0.35	0.56	0	0

Now, we add the S_R to the context and build the Lattice to ensure returning relevant results. We fix the threshold θ at 0.45. Table 11 presents the new context builder which will be transmitted to the Lattice Builder.
The Relevant Service Selector is charged then to identify most relevant service to S_R using appropriate algorithms for browsing Lattices. In our case only and according to the generated lattice in Fig. 3 three operations are selected and then transmitted to the Reasoning Module. Those operations are $\{op_{11}$, op_{31}, $op_{21}\}$. Thus, the composition process is reduced to reason only on three operations in which op_{31} can be composed with op_{11} or op_{21}.

The example illustrates the advantage of the Service Classification Module in anticipating the composition task, maintaining the composition plan

Table 10. The operation SimCxt for $\theta = 0.45$

	op_{11}	op_{21}	op_{22}	op_{31}	op_{41}	op_{51}	op_R
op_{11}	X	X					
op_{21}	X	X					
op_{22}			X				
op_{31}				X			
op_{41}					X	X	
op_{51}					X	X	
op_R	X	X		X			X

Table 11. The service SimCxt for $\theta = 0.45$

	op_{11}	op_{21}	op_{22}	op_{31}	op_{41}	op_{51}	op_R
WS_1	X	X					
WS_2	X	X	X				
WS_3				X			
WS_4					X	X	
WS_5					X	X	
S_R	X	X		X			X

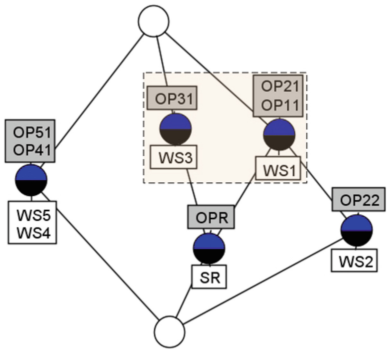

Fig. 3. The generated lattice including S_R

working as much as possible by notifying the reasoner that op_{11} and op_{21} are similar.

2.3 Service Reasoning Module

The Service Reasoning module identifies the candidate composition plans that realize the goal (S_R) based on a logic reasoner. IDECSE relies on a logic Reasoner that takes into account results given from the Service Classification Module.

In order to enable working with large collections of Web services, IDECSE distinguish between Web services types and instances. A Web service type is a set of Web service instances with identical functionality. A Web service type is semantically described by the inputs, outputs, preconditions and effects that capture the functionality offered by this type of services. Web services instances are the actual services that can be invoked. This separation allows reducing the search space, which helps ensuring the scalability of the composition process [16]. The main components of the Service Reasoning Module are:

- Reasoner: is responsible for checking the ontology consistency in addition to handling the maintenance of the state including preconditions and effects application.
- Filter: avoids redundancy from the plan by identifying service types with potential relevance to the goal and checks the dependency relationships between each two consecutive service types.
- Matchmaker: allows querying the Service Registry for available services in order to match the preconditions of a Web service with the effects of another service.
- Abstract planner: can be considered as the main component of this module and is responsible for generating a set of abstract plans.

IDECSE relies on a planning algorithm [1] to create a composition of the available service types by providing an abstract plan that meets the functional requirements. In this algorithm, inputs and outputs are distinguished conditions where inputs are attempted to be achieved before preconditions since preconditions may have arguments consisting of several inputs. A plan of composite services is formalized as a proof of the goal to answer the user query. Thus, a Plan $\{\mathcal{P}\} = \{A_i\}_{i=1..n}$ is a sequence of n actions A_i. Each action applies on a state E_i to produce a state E_{i+1}: $\forall i \in \{0, .., n-1\}, E_i \wedge A_i \models E_{i+1}$. Starting from an initial state E_0 the plan produces the goal G: $E_0 \wedge \{\mathcal{P}\} \models G$. Therefore composite services generated are new services that meet all the query requirements. Finally, generated plans are transmitted to the service execution module.

2.4 Service Execution Module

The Service Execution Module translates the abstract plan into an executable one by associating to each service type its specific instances using the Service Instances Registry. The plan generated by the previous module is considered as a template for the composite service and drives the process of matching each service type to a corresponding service instance. The Service Execution Module is mainly composed of the two following components:

- The Executable Plan Generator considers non-functional requirements of the goal (provided by Service Request Module) and enables to concretize the abstract plan generated by the abstract Reasoning Module.
- The Executable Composition Analyzer generates executable code and invokes the execution engine.

In IDECSE, the Executable Plan Generator implementation is based on the algorithm presented in [17]. This algorithm takes as input a composition plan, the QoS permissible values imposed by the user, and their weights and generates as output a composition plan that satisfies the requirements of the user.

2.5 Service Monitoring Module

Monitoring deals with the actual execution of the composite service and is responsible for monitoring the execution and recording violation of any requirement of the goal service at run-time. For this purpose, we plan to propose an event-based monitoring framework that allows specifying and reasoning about the monitoring properties during composition process execution. Properties to be monitored are specified by the user and added to process specification at both design and execution time. These properties include functional constraints (invocation and execution order), non-functional properties (security, QoS...), temporal constraints (response time, invocations delay), data constraints (data availability, validity and expiry). These properties can be also combined such as monitoring the data validity and access control within specific time frame.

Services providers can also specify additional assumptions about the composition process in terms of events extracted from its specification. Run-time deviations and inconsistencies will be monitored by using a variant of techniques developed for checking integrity constraints in temporal deductive databases. When requirements variations are detected, a deviation notification is sent to the composition manager. This notification indicates: (i) the requirement that has been violated, (ii) the malfunctioning service(s) that violated it, and (iii) diagnostic information regarding the violation. Based on the type of the deviation notification and service monitoring policies, recovery policies will be triggered.

3 Implementation and Validation

The overall architecture for the IDECSE approach is under development as a java application that presents all the functionalities from design phase to monitoring and recovery phases. This application contains a user friendly interface to specify the composition design (services/constraints/ and control/ data flow), to automatically generate the corresponding models, to invoke the reasoner, and show the results returned. The application provides also an assumption editor to specify assumptions and check their correctness.

Once each type in the selected plan is bound to a concrete Web service instance, a generator produces a concrete workflow that can be deployed onto a runtime infrastructure, to realize the composite service. For that, we first generate the WSDL description (name, interface, port types) for the composite service. Then, we define partner link types to link the component services, and proceed to the generation of the composition flow (BPEL flow for instance). The selected plan gives the invocation order. We use an Eclipse Modeling Framework (EMF) model of BPEL (WSDL) that is automatically created from a BPEL

(WSDL) schema. The model provides in-memory representation of constructs and support for persistence to files (serialization) and loading from files (deserialization). BPEL and WSDL manipulation become significantly simplified with the corresponding EMF models. In case of conflicts, the monitoring process is initiated by the monitoring manager after receiving a request to start a monitoring activity as specified by a monitoring policy. First, it checks if the requested constraint or property can be monitored or not. This checking is based on the composition process identified in the policy, and the event reporting capabilities indicated by the type of the execution environment of the composition process. If the requested constraint can be monitored, the monitoring manager triggers an event listener to capture events from the composition execution environment and passes to it the events that should be collected. It also sends to the monitor the specification of the constraint to be checked.

These components are still under refinement and tests. Then, we plan to compare our results with important approaches close to our contribution such as [18]. Also, it will be interesting to measure the performance of the IDECSE framework before and after incorporating the similarity based approach in order to show its added value. To conduct fair experiments, we need a sufficient number of services and ontologies with a variety of sizes. However, it is very hard to collect or manually construct appropriate data. For this reason, we may randomly generate experimental data.

4 Related Work

A considerable number of research efforts have focused on various aspects of Web service compositions ranging from semantic service discovery to service specification, composition, deployment, and monitoring [19–25]. A deep analysis of these efforts shows that some has focused on the execution aspects of composite Web services by considering WSDL to describe Web services and BPEL to compose them, without much consideration for requirements that drive the development process. Some other efforts are concerned by the feasibility of the composition process by considering semantics and AI approaches without taking into account the run-time deployment and execution. However, we believe that these two approaches can be complementary and can be combined for managing the global life-cycle of the composition i.e. specifying, composing, verifying, deploying, monitoring, and analyzing to achieve a full governance of the composition. Semantic classification of services was also investigated. For instance, [26] uses FCA formalism to highlight the relationships between services and permits the identication of different categorizations of a certain service. Lattice of services is built in [27], by extracting keywords from their specification. [28] combines text mining and machine learning techniques for classifying services. This approach improves the performance of Web-service clustering by considering the two main steps in the clustering process. It introduces first a Web-service similarity-measuring approach that uses both ontology learning and IR-based term similarity. Second, it proposes an approach to identifying cluster centers by using similarity values for service names to improve the performance in a second step. Using ant-based algorithm, [29] considers the degree of semantic similarity between services as the main clustering criterion. The semantic description is based on measuring similarity between ontological concepts referred by input and output parameters of the service. A matching method and metrics are used to measure the semantic similarity. Other approaches rely on measuring similarity between service, [13] proposes a novel approach for Web service retrieval based on measuring similarity between services interfaces. The evaluation of the similarity between Web services considers both the semantic and the structure of a WSDL description with a semantic annotation.

5 Conclusion and Future Work

This paper describes IDECSE, a new semantic integrated approach for composite services engineering. Compared to existing approaches, IDECSE considers semantics in all the composition global life-cycle, addresses the challenge of fully automating the composition processes, and proposes and adapts reasoning, monitoring, and adaptation techniques. The main new added values are on the Web services classification level. In fact, we propose to enhance IDECSE with semantic measures and FCA for building Web service Lattices according to functionality domains. Similarity measures are used to calculate semantic and syntactic similarity based on OWL-S. Our work in progress includes the enrichment of service Lattices with QoS aspects and user preferences. We also plan to extend the framework to include additional features such as failure handling, and an interactive visual environment for testing composite services.

References

1. Rouached, M., Messai, N.: SCoME: a web services composition modeling and engineering framework. In: 2013 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT), IEEE (2013)
2. Abid, A., Messai, N., Rouached, M., Devogele, T., Abid, M.: IDECSE: a semantic integrated development environment for composite services engineering
3. Lara, R., Roman, D., Polleres, A., Fensel, D.: A conceptual comparison of WSMO and OWL-S. In: (LJ) Zhang, L.-J., Jeckle, M. (eds.) ECOWS 2004. LNCS, vol. 3250, pp. 254–269. Springer, Heidelberg (2004)
4. Martin, D., Burstein, M., Hobbs, J., Lassila, O., McDermott, D., McIlraith, S., Narayanan, S., Paolucci, M., Parsia, B., Payne, T., et al.: OWL-S: semantic markup for web services. W3C member submission, vol. 22 (2004)
5. Rada, R., Mili, H., Bicknell, E., Blettner, M.: Development and application of a metric on semantic nets. IEEE Trans. Syst. Man Cybern. **19**, 17–30 (1989)
6. Wu, Z., Palmer, M.: Verbs semantics and lexical selection. In: Proceedings of the 32nd Annual Meeting on Association for Computational Linguistics, Association for Computational Linguistics (1994)
7. Tversky, A.: Features of similarity. Psychol. Rev. **84**, 327–352 (1977)
8. Resnik, P.: Using information content to evaluate semantic similarity in a taxonomy. arXiv preprint cmp-lg/9511007 (1995)
9. Jiang, J.J., Conrath, D.W.: Semantic similarity based on corpus statistics and lexical taxonomy. arXiv preprint cmp-lg/9709008 (1997)

10. Rubenstein, H., Goodenough, J.B.: Contextual correlates of synonymy. Commun. ACM **8**, 627–633 (1965)
11. Miller, G.A., Charles, W.G.: Contextual correlates of semantic similarity. Lang. cogn. process. **6**, 1–28 (1991)
12. Sánchez, D., Batet, M., Isern, D., Valls, A.: Ontology-based semantic similarity: a new feature-based approach. Expert Syst. Appl. **39**, 7718–7728 (2012)
13. Plebani, P., Pernici, B.: URBE: web service retrieval based on similarity evaluation. IEEE Trans. Knowl. Data Eng. **21**, 1629–1642 (2009)
14. Ganter, B., Wille, R.: Formal Concept Analysis: Mathematical Foundations. Springer, New York (1999)
15. Azmeh, Z., Hamoui, F., Huchard, M., Messai, N., Tibermacine, C., Urtado, C., Vauttier, S.: Backing composite web services using formal concept analysis. In: Jäschke, R. (ed.) ICFCA 2011. LNCS, vol. 6628, pp. 26–41. Springer, Heidelberg (2011)
16. Agarwal, V., Chafle, G., Dasgupta, K., Karnik, N., Kumar, A., Mittal, S., Srivastava, B.: Synthy: a system for end to end composition of web services. Web Semant. Sci. Serv. Agents World Wide Web **3**, 311–339 (2011)
17. Ko, J.M., Kim, C.O., Kwon, I.H.: Quality-of-service oriented web service composition algorithm and planning architecture. J. Syst. Softw. **81**, 2079–2090 (2008)
18. Cugola, G., Ghezzi, C., Pinto, L.S.: DSOL: a declarative approach to self-adaptive service orchestrations. Comput. **94**(7), 579–617 (2012)
19. Xiaoming, P., Qiqing, F., Yahui, H., Bingjian, Z.: A user requirements oriented dynamic web service composition framework. In: Proceedings of the 2009 International Forum on Information Technology and Applications, IFITA 2009, vol. 1, pp. 173–177. IEEE Computer Society, Washington, DC (2009)
20. Hatzi, O., Vrakas, D., Nikolaidou, M., Bassiliades, N., Anagnostopoulos, D., Vlahavas, I.: An integrated approach to automated semantic web service composition through planning. IEEE Trans. Serv. Comput. **5**(3), 319–332 (2012)
21. Marconi, A., Pistore, M.: Synthesis and composition of web services. In: Bernardo, M., Padovani, L., Zavattaro, G. (eds.) SFM 2009. LNCS, vol. 5569, pp. 89–157. Springer, Heidelberg (2009)
22. Rabah, S., Ni, D., Jahanshahi, P., Guzman, L.F.: Current state and challenges of automatic planning in web service composition. CoRR (2011)
23. Kuang, L., Li, Y., Wu, J., Deng, S., Wu, Z.: Inverted indexing for composition-oriented service discovery. In: 2007 IEEE International Conference on Web Services, Salt Like City, USA, pp. 257–264 (2007)
24. Kona, S., Bansal, A., Gupta, G.: Automatic composition of semantic web services. In: ICWS, pp. 150–158 (2007)
25. Lecue, F., Mehandjiev, N.: Towards scalability of quality driven semantic web service composition. In: Proceedings of the 2009 IEEE International Conference on Web Services, ICWS 2009, pp. 469–476. IEEE Computer Society, Washington, DC (2009)
26. Aversano, L., Bruno, M., Canfora, G., Di Penta, M., Distante, D.: Using concept lattices to support service selection. Int. J. Web Serv. Res. (IJWSR) **3**, 32–51 (2006)
27. Bruno, M., Canfora, G., Di Penta, M., Scognamiglio, R.: An approach to support web service classification and annotation. In: Proceedings of the 2005 IEEE International Conference on e-Technology, e-Commerce and e-Service, EEE 2005. IEEE (2005)

28. Kumara, B.T., Paik, I., Chen, W.: Web-service clustering with a hybrid of ontology learning and information-retrieval-based term similarity. In: 2013 IEEE 20th International Conference on Web Services (ICWS). IEEE (2013)
29. Pop, C.B., Chifu, V.R., Salomie, I., Dinsoreanu, M., David, T., Acretoaie, V.: Semantic web service clustering for efficient discovery using an ant-based method. In: Essaaidi, M., Malgeri, M., Badica, C. (eds.) Intelligent Distributed Computing IV. SCI, vol. 315, pp. 23–33. Springer, Heidelberg (2010)