Service Composition: State of the art and future challenges

G. Kapitsaki, D.A. Kateros, I.E. Foukarakis, G. N. Prezerakos, D.I. Kaklamani and I.S. Venieris

Abstract — **Service discovery and service composition constitute challenging tasks of service provisioning. Service composition is especially gaining on importance, as it can produce new composite services with features not present in the individual services. Many methods have been proposed over the years to address these issues. Simple Mobile Services (SMS) is a project aiming at creating innovative tools that will introduce a new class of mobile services enabling individuals and small businesses to become service providers. In this paper an attempt is made to present an overview of all existing techniques on service composition and examine them in the context of SMS. The enumeration of the techniques is complete and focuses mostly on recent approaches. Moreover, service composition is discussed under the prism of mass service provisioning by mobile operators where some techniques currently seem to be more suitable than others.**

I. Introduction

Service composition can be defined as the process of combining and linking existing services (atomic or composite) to create new working services. It constitutes an essential part of service provisioning, since it leads to novel service offering thus adding value that was not existent in the individual services. The Web services paradigm has currently gained momentum as a service design and deployment architecture therefore most service composition approaches concentrate on Web service composition.

The aim of Simple Mobile Services (SMS) project [1] is to create innovative tools enabling a new class of services, addressing the specific needs of mobile users and enabling individuals and small businesses to become service providers. In this context, a scalable service composition scheme is envisaged. In this paper we investigate existing methodologies of service composition discussed in the literature, in order to assess suitable techniques to be exploited within the scope of the SMS project.

Manuscript received January 31, 2007. This work was supported in part by the European Commission. This work has been performed in the framework of the European funded project SMS. The project has received research funding from the Community's Sixth Framework programme. The authors would like to acknowledge the contributions of their colleagues from the SMS consortium. The views expressed in this document do not necessarily represent the views from the complete SMS consortium. The Community is not liable for any use that may be made of the information contained therein.

G. Kapitsaki, D.A. Kateros, I.E. Foukarakis G. N. Prezerakos, D.I. Kaklamani and I.S. Venieris are with the National Technical University of Athens, School of Electrical & Computer Engineering (e-mail: {gkapi, dkateros,ifouk}@mail.ntua.gr, prezerak@central.ntua.gr {venieris,dkaklam}@cs.ntua.gr).

In order to create executable compositions of existing Web services, the use of composition languages has been suggested. Using such languages the composition can be expressed as a model that can be run by an execution engine. The Business Process Execution Language (BPEL) is probably the most promising composition language. BPEL defines business processes that interact with external entities. It has adopted Web services as its external communication mechanism and the use of WSDL 1.1 to describe outgoing and incoming messages. The latest version of BPEL, WS-BPEL 2.0 [2] is currently being standardized within the OASIS by the WSBPEL Technical Committee. Other notable composition language proposals include the Web Service Conversation Language (WSCL) [3], which defines the overall input and output message sequences for one Web service using a finite state automaton over the alphabet of message types and the Web Service Choreography Interface (WSCI) [4], an XML-based language, which describes interfaces used to specify the flow of messages at interacting Web services, both specified by the W3C.

Regarding service composition methods different techniques can be found in the bibliography. These techniques are usually divided into the categories of static (services to be composed decided at design time) and dynamic composition (services to be composed decided at runtime), or automatic (no user intervention) and manual composition (user-driven composition). In this paper the service composition approaches are categorized according to the research area they base their methods on (Figure 1): AI planning introduces the principles of Artificial Intelligence into service composition, Semantic Web approaches exploits the semantic characteristics of a service and Middleware approaches consider different middleware solutions.

The paper is organized as follows: in the next section the most important AI planning techniques are presented. Section III introduces semantic Web approaches of service composition, while section IV focuses on middleware approaches. Some other solutions proposed are presented in section V and section VI concludes the paper.

II. AI Planning

Web service composition can be seen as the construction of a process to attain a certain goal. This is a problem which has been investigated extensively by research in Artificial Intelligence (AI). A classical planning problem includes a description of the initial state of the world, a description of the

desired goal, and a description of the possible actions which may be executed. In this section of the paper we make an overview of AI approaches to planning that have been investigated for the purpose of Web service composition.

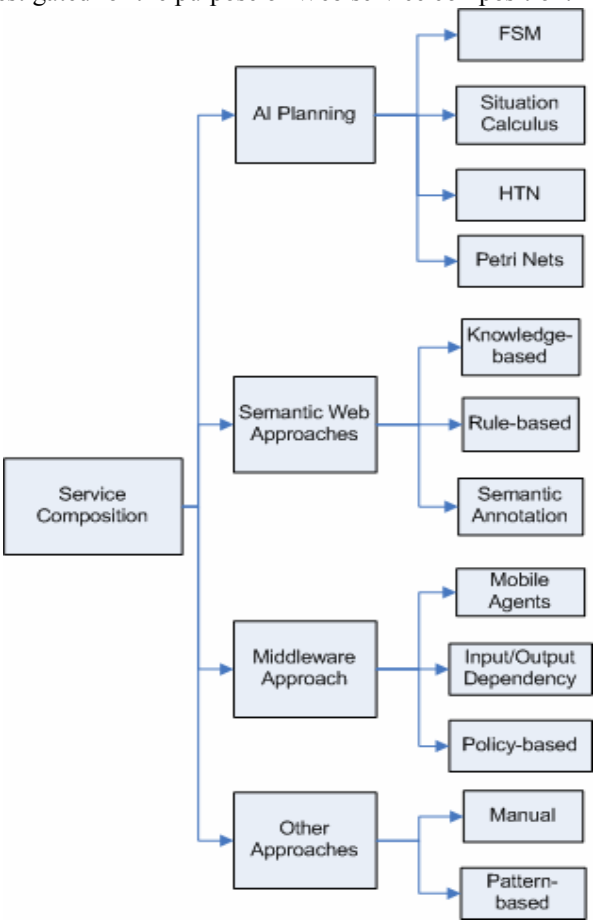

Figure 1 – Service Composition Categorization

A. *Finite State Machines*

A finite state machine (FSM) is a model of behavior composed of a finite number of states, transitions between those states, and actions. In [9] the authors focus on Web services whose schema (both internal and external) can be represented using a FSM and propose an algorithm that checks a composition's existence, and returns one if it exists. A specific type of FSM, that generates an output based on its current state and additionally an input, Mealy Machines, have been proposed for service modeling in [10]. According to this approach services communicate by sending asynchronous messages, and each service has a queue. A global "watcher" keeps track of all messages. The conversation is introduced as a sequence of messages. By studying and understanding conversation properties, the method provides new approaches for designing and analyzing "well-formed" service composition.

B. *Situation calculus*

In Situation Calculus a dynamic world is modeled as progressing through a series of situations as a result of various actions being performed within the world. A situation represents a history of action occurrences. The constant S_0 describes the initial situation where no actions have occurred yet. The transition to the successor situation of s as result of an action a is denoted using $do(a,s)$. Lastly, the truth value of statements given a situation is modeled by fluents which are denoted by predicate symbols and take a situation as their last argument. In [5] the authors argue that composition of Web services, viewed as execution trees, can be realized, at least under certain assumptions, in reasoning about actions formalisms, making specific use of Situation Calculus. In [6] the use of an augmented version of Golog is suggested. Golog is a high-level logic programming language, built on top of the Situation Calculus, for the specification and execution of complex actions in dynamical domains. The authors extended Golog to provide knowledge and sensing actions in order to become a suitable formalism for representing and reasoning about the semantic Web services composition problem. The general idea of the method described is that software agents could reason about Web services to perform automatic Web service discovery, execution, composition and inter-operation.

C. *Hierarchical Task Networks*

Hierarchical Task Network (HTN) [7] planning is a method of planning by task decomposition. In contrary to the other concepts of planning, the central concept of HTNs is not states, but tasks. An HTN based planning system decomposes iteratively the desired task into a set of sub-tasks until the resulting set of tasks consists only of atomic (primitive) tasks, which can be executed directly by invoking some atomic operations. During each iteration of task decomposition, it is tested whether certain given conditions are violated (e.g. exceeding a certain amount of resources) or not. The planning problem is successfully solved, if the desired complex task is decomposed into a set of primitive tasks without violating any of the given conditions. An approach of using HTN planning in the realm of Web Services was proposed in [8], facilitating the planning system SHOP2, which uses DAML-S Web service descriptions for automatic service composition.

D. *Petri Nets*

A Petri net is a directed, connected, and bipartite graph in which nodes represent places and transitions, and tokens occupy places. When there is at least one token in every place connected to a transition, that transition is enabled. An enabled transition might fire by removing one token from every input place, and depositing one token in each output place. In [11] the authors model services as Petri nets by assigning transitions to methods and places to states. Each service has an associated Petri net that describes service behavior and has two ports: one input place and one output place. At any given time, a service can be in one of the not instantiated, ready, running, suspended, or completed states. After the definition of a net for each service, composition operators (sequence, alternative, unordered sequence, iteration, parallel with communication, discriminator, selection, and refinement) perform composition.

III. Semantic Web Approaches

Composition proposals of this category use the principles of the Semantic Web to provide more concrete service

composition by considering the semantic properties of services. While other methods identify only the structure of the messages exchanged, semantic approaches also interpret the message content.

A. Semantic annotation

Semantic annotation marks up the Web services description with metadata that correspond to ontologies. Markup languages, such as DAML+OIL and OWL, are used for publishing and sharing data using ontologies. Semantically enriched descriptions for Web services are available through OWL-S (formerly DAML-S). OWL-S uses Service Profile, Process Model and Grounding to describe a Web service. The Service Profile provides the information needed for an agent to discover a service, while the Process Model and Grounding provide enough information for an agent to make use of a service, once found.

The AI solution of [6] introduces semantic markup in DAML-S. METEOR-S initiative describes a framework for semi-automatic annotation of Web services (MWSAF) [12]. It proposes several algorithms used in matching XML schemas to ontologies. The linguistic and structured similarity of concepts is examined and then the best mapping between the matches is found. The architecture is composed of three components: the Ontology-Store stores the existing ontologies, the Translator Library performs the transformation to SchemaGraph (common representation for XML Schema and ontologies description) and the Matcher Library provides the different matching algorithms. WSDL-S, a mechanism for annotating Web services with semantics, derived from METEOR-S and is now the basis for the Semantic Annotations for WSDL (SAWSDL) [13]. SAWSDL describes how to add semantic annotations to various parts of a WSDL document.

B. Rule based approaches

Other approaches for semantic composition define specific rules that guide the service composition process. In [14] a composability model that checks which Web services can interact with each other based on composability rules that check the syntactic and semantic properties of the services is described. Based on this compasibility model a four phase approach for automatic service composition is proposed: the Specification phase enables high-level description of composite services through the Composite Service Specification Language (CSSL), the Matchmaking phase uses compasibility rules to generate composition plans according to the specifications, Selection phase selects the best plan and Generation phase provides a description of the composite service in a chosen composition language. In [15] policy compatibility is introduced into service composition. Services are composed based on service policies imposed by organisations/component providers, service flow policies associated with the complete service workflow and user policies that represent user requirements. Syntactic and semantic compatibility is also taken into account.

C. Knowledge based composition

Knowledge based composition introduces domain-specific knowledge into the service composition process. This approach is based on the model of knowledge-based decision support systems that make extensive use of domain knowledge. In [16] domain-specific knowledge is used to guide the service composition. It uses ontologies as knowledge models with a commonly accepted vocabulary in order to be able to reuse the knowledge-based advice systems within the service oriented framework of the Grid computing. The composition framework proposed exploits domain-specific knowledge and provides guidance to compose Web/Grid services into a workflow specification. The user gives an initial description of the problem and the system offers advice on services to compose.

IV. Middleware Approach

The appearance of Service Oriented Architectures is the result of the need to expose functionalities and access to resources available from different organizations. An important part of these applications is the service composition mechanisms that enable the applications to become adaptable, reconfigurable and fault-tolerant. Many different approaches have been used to solve this problem, but most of them rely on middleware that enables discovery and invocation of services. Although many different types of middleware that enable service composition exist, they share two common parts. The first one is a registry containing information about existing services, such as location, inputs and outputs. The second part is the engine that actually composes the distinct services and is responsible for running the application.

A. Mobile Agents

The first approaches for service composition can be found in the mobile agent technology. Mobile agents are autonomous entities that try to achieve a goal either on their own or by collaborating. One of their advantages is that they can use semantic language to support complex interactions. As described in [17], mobile agent applications were initially based on components, but the abstraction that the semantic languages offer enables the interaction between mobile agents and services. Agents can discover the services they need to invoke either by searching a registry or by asking other agents. Also rules that can guide the behavior of the agents can be defined by the user. The main benefit of this approach is that agents are not limited to the execution of a workflow, but can follow the user and his actions and react to changes.

B. Input/Output Dependency

A different approach is described in [18]. The interesting thing in this approach is the way dynamic discovery of services is achieved. Each service is described by some keywords, the list of its inputs and the list of its outputs. Each output can be linked to zero, one or a list of inputs, so that input requirements to produce a specific output are described. The process of service composition starts by creating lists of services that have similar inputs and produce similar outputs.

If a simple service is not found, then the search continues to create a list of services that take as input some of the initial inputs and possibly the outputs of the services in the previous list. Then an invocation graph is created. If a service fails or needs to be replaced, then the system uses the list of the input-output dependencies to find a replacement.

C. Policy Based Approaches

The problem of ensuring that policies are obeyed in different contexts is discussed in [19]. A service composition layer is proposed that takes as inputs the policies and the domain knowledge of the application. Interfaces to services are defined, enabling in this way interoperability with many different service discovery paradigms (UPnP, JINI etc) and communication protocols (CORBA, DCOM, RMI etc). These interfaces can be generated easily when they are already described in a language such as WSDL or OWL-S, but can also be written by a user. A language with a Java-like syntax is used for this purpose. The main element of this language is the binding variables, that provide links to remote services depending on their type and their description attributes. New objects or instances are not created directly through the language, but either by imported factory methods or by the runtime context. The usage of the strict syntax of this language ensures that the final composed service is obeying the policies of the current context.

V. Other Approaches

A. Composition based on patterns

Patterns can be defined as sets of generic rules which can be used to make or generate a solution to a problem. In [20] the use of patterns in service composition is introduced. Assuming that patterns can be used to represent reusable business process logic, the use of patterns, particularly in the representation of non-functional requirements, is illustrated through a payment mechanism example. The use of patterns for the coordination of services from different participants from a workflow-based viewpoint is proposed in [21]. As participants are dynamically changing when realising application services, the abstract interaction logic of services becomes concrete only at provision time. Different coordination alternatives can be realised at that point. This work describes the use of generic interaction patterns at design time that serve as the basis for the analysis and optimisation of the interaction logic at runtime. Apart from the use of software patterns, another category of patterns, namely usage patterns, can play an important role. This approach has been investigated in [22] where a collaborative filtering system is used for processing usage patterns comprised of past user decisions that have been saved together with context information about the user at the time the decision was made. Based on this information, it is possible to predict how users with similar profiles will behave in the same situation. In a similar fashion it would be possible to use statistical analysis over large sets of such service usage patterns (e.g. inside a mobile operator) to automatically derive sets of services that are used, by subscribers with similar profiles and under similar contexts, towards a certain task. These services are, by definition, ideal candidates for composition into a new service that accomplishes this task.

B. Manual service composition based on modeling

A more straightforward approach is presented in [23]. The authors suggest the use of UML as the integration platform for Web service composition, as UML models provide Web service representation in an abstract level and are easier to understand by humans. The method utilizes a two-way mapping between a WSDL service description and a WSDL-independent service model in UML. This mapping allows automating the transformation of WSDL descriptions to UML models and vice versa. The complete composition procedure is outlined in [24] and consists of four distinct steps: a) The target composite Web service is modeled using two submodels; a UML class diagram to model its interfaces and a UML activity diagram to model the composition and workflow. Then the developer chooses suitable existing component Web services to create the target service. b) The activity diagram of the composed Web service is created. This is achieved by extracting the WSDL description of the component Web services and transforming it to UML class diagrams. The extracted information is represented as tagged values according to a UML profile. This profile allows control and data flows to be modeled. The activity model of the composite service results by mapping operations of the existing component services to the activities included in the activity model of the composite Web service. c) Using the activity diagram and appropriate transformations the needed XML documents for configuring the execution environment are generated d) Finally, the service is deployed and published.

The weight of this approach falls into the discovery of appropriate services. Once the services have been discovered then composition into the desired services takes place manually, via UML modeling. However it is a straightforward task and allows a great degree of flexibility in transforming one service's results into the next service's input data.

VI. Discussion and Challenges

The methods presented above take into account different characteristics of the services to provide more sophisticated service composition, use different representation mechanisms and degree of automatic composition. However, there are certain composition requirements that apply to all techniques. The goal of the service composition is the combination of services in such a way that they can interact successfully with each other and result in providing a complete and workable composite service. During composition both functional and non-functional service characteristics have to be taken into account in order to provide a complete solution.

Within the scope of the SMS project, the developer must be able to create complex services reusing pre-built service components. Interaction between the components should be described in workflow-like manner, utilizing the functionality

provided by the components. The overall process must produce code that can be executed in a runtime environment. However, the intervention of the developer in code level must be as limited as possible, and changes in the process must occur mostly in the modeling level. More information on the creation of SMS services can be found in [25] and [26].

AI methods are suitable for modeling the inter-workings of a complex service. Although the ideas expressed within the various AI methodologies are very promising, they are difficult to apply in real world environments. They also do not have a way of expressing important metadata related to the service components.

Semantic approaches examine the service compatibility, take into account both functional and non-functional requirements and allow reasoning on which component is best to use in each situation. They can easily find alternates to existing services and replace them if needed. However they are designed to allow description of services in a machine-readable way, making it difficult to be used by humans.

The various middleware solutions aim to combine the above paradigms. They use AI methods to model the composed service flow, while they take advantage of semantic description to allow interoperability between services. This functionality is transparent to the developer. They also provide human-understandable forms of the composed services that allow the developer to customize his solution in code level. However they do not take into account the requirement for building a service model prior to the actual service development in software.

Also all the above approaches ignore the business side of providing composite services. In practice, operators will have to set up and manage a variety of business agreements, sometime quite complex, that will enable them to access and exploit component services offered by 3rd parties. Under this perspective, automation in service composition is a much more complicated issue than the ability to provide a composition algorithm with a satisfying performance.

Along this line, manual service composition techniques in conjunction with service usage patterns seem to be promising approaches for a commercial deployment of service composition activities in a telecom operator. Analysis of service usage patterns can identify sets of services which are good candidates for the creation of new composite services. Consequently a manual process can initiate the conversion of these services into UML models and the creation of a composite service model that in the end is transformed into a new executable composite service.

It is evident that we are still far from achieving fully automated service composition. However, the process can be automated to a large extent and then fine tuned with a moderate amount of human intervention. In addition, the adoption of standards in service description and service modeling will allow the majority of developers to use already existing development environments.

VII. References

[1] Simple Mobile Services – http://www.ist-sms.org

[2] OASIS WSBPEL TC, "Web Services Business Process Execution Language Version 2.0 Public Review Draft", August 2006

[3] W3C, "Web Service Conversation Language (WSCL) 1.0", 2002

[4] W3C, "Web Service Choreography Interface (WSCI) 1.0", 2002

[5] D. Berardi et al., "Reasoning about Actions for e-Service Composition", ICAPS Workshop on Planning for Web Services (P4WS 2003), 2003

[6] S. McIlraith and T. Son, "Adapting Golog for Composition of Semantic Web Services", KR '02, 2002

[7] K. Erol, J. Hendler and D. S. Nau, "Semantics for Hierarchical Task Network Planning", UMIACS Technical Report, 1994.

[8] D. Wu, E. Sirin, J. Hendler, D. Nau, and B. Parsia, "Automatic Web Services Composition Using SHOP2", Workshop on Planning for Web Services, 2003.

[9] D. Berardi et al., "Automatic Composition of E-Services that Export Their Behavior," Proc. 1st Int'l Conf. Service-Oriented Computing (ICSOC 03), LNCS 2910, Springer-Verlag, 2003, pp. 43–58.

[10] T. Bultan et al., "Conversation Specification: A New Approach to Design and Analysis of E-Service Composition," Proc. Int'l World Wide Web Conf. (WWW 2003), ACM Press, 2003, pp. 403–410.

[11] R. Hamadi and B. Benatallah, "A Petri-Net-Based Model for Web Service Composition," Proc. 14th Australasian Database Conf. Database Technologies, ACM Press, 2003, pp. 191–200.

[12] A. Patil, S. Oundhakar, A. Sheth, K. Verma, "METEOR-S Web Service Annotation Framework", The Proceedings of the Thirteenth International World Wide Web Conference, May 2004 (WWW2004)

[13] Semantic Annotations for Web Services Description Language, http://www.w3.org/2002/ws/sawsdl/

[14] B. Medjahed, A. Bouguettaya, A. K. Elmagarmid, "Composing Web Services on the Semantic Web", The VLDB journal, September 2003

[15] S. A. Chun, V. Atluri, N. R. Adam, "Using Semantics for Policy-Based Web Service Composition, Distributed and Parallel Databases", 2005

[16] C.hen L., Shadbolt N.R., Goble C., Tao F., Cox S.J., Puleston C., Smart P.R., "Towards a Knowledge-based Approach to Semantic Service Composition", In Proceedings of 2nd International Semantic Web Conference (ISWC2003)

[17] M. Vallee, F. Ramparany, L. Vercouter, "A Multi-Agent System for Dynamic Service Composition in Ambient Intelligence Environments"

[18] W. Zahreddine, Q. H. Mahmoud, "A Framework for Automatic and Dynamic Composition of Personalized Web Services"

[19] J. Robinson, I. Wakerman, T. Owen, "Scooby: Middleware for Service Composition in Pervasive Computing"

[20] M. T. Tut, D. Edmond, "The use of Patterns in Service Composition", 2002

[21] C. Zirpins, W. Lamersdorf, T. Bauer, "Flexible Coordination of Service Interaction Patterns", (ICSOC04)

[22] A. Chen, "Context-aware collaborative filtering system: Predicting the user's preference in the ubiquitous computing environment," in LoCA, 2005, pp. 244–253.

[23] R. Grønmo, D. Skogan, I. Solheim, and J. Oldevik, "Model-driven Web Services Development," EEE-04, Taipei, Taiwan, 2004

[24] D. Skogan, R. Grønmo, I. Solheim, "Web Service Composition in UML", EDOC-04, Monterey, Canada, 2004

[25] G. Bartolomeo, N. Blefari Melazzi, G. Cortese, A . Friday, G. Prezerakos, S. Salsano, R. Walker, "SMS: Simplifying Mobile Services for Users and Service Providers", AICT/ICIW 2006, Guadeloupe, French Caribbean

[26] G. N. Prezerakos, N. G. Tselikas and G. Cortese, "Model-driven Composition of Context-aware Web Services Using ContextUML and Aspects", ICWS 2007, Salt Lake, Utah, USA