Chapter XII
Web Services Automation

L. Koumakis
Technical University of Crete, Greece
Foundation for Research and Technology–Hellas (FORTH), Greece

V. Moustakis
Technical University of Crete, Greece
Foundation for Research and Technology–Hellas (FORTH), Greece

G. Potamias
Foundation for Research and Technology–Hellas (FORTH), Greece

ABSTRACT

Web services revolutionize use of information systems and to some extent academic research. Put it simply, web services come in small pieces of software, which can be put together to address complex tasks over heterogeneous and distribute data sources. In doing so, services represent an effective solution to isolated software integration. Web services are based on standards; HTML is used as communication protocol layer and XML is used as basic language. Web service composition and orchestration is not trivial. Representation of a human inquiry to an efficient orchestra of services must pass from a translation step, made possible via the use of metadata, which carry real world semantics. To this end, semantic web services denote intention to provide additional information about, and to facilitate, individual service integration and automatic composition. The chapter presents a concrete methodology to support the use of automatic composition of complex semantic web services with natural language. Web service semantics are linked with natural language processing capabilities to empower users to write descriptions in their own language and in the sequel to have these descriptions mapped automatically into a well tuned web service orchestra.
INTRODUCTION

A web service (WS) is defined by the W3C as “a software system designed to support interoperable Machine to Machine interaction over a network. Web services add a new chapter to the success story of semantic web standards. Open standards, like XML, foster universal exchange of information over internet. WS technology adds universal interchange ability and thus universal availability of application logic. Web services represent the second generation of internet tools to connect people to things they are dealing with. They are not connecting people with html web pages; they are connecting their business applications with those of their colleagues, customers, partners and suppliers. Web services could in fact revolutionize the way we develop applications like the internet itself changed our life.

The use of standards has come up as essential part from the first steps of WS technology. Standards are pre-requisite for interoperability. Users want their web services to link and interact with those of their partners and colleagues in a standardized way, yet, personalized to their needs and preferences. The launch of XML opens new avenues for a completely new type of interoperability of software across networks. The desire to use each other’s applications in order to develop new ad-hoc services and appliances is within reach.

Unfortunately semantic web standards cannot solve all the problems that appear in web services. Even though they support interoperability, the development of large and complex domain-specific applications still remains complicated and time consuming.

Interchange availability of web services does not signal automatic pathway to new horizons in designing IT-based business processes. Real applications emerge from a complex and dynamic composition of a number of web services. Almost all business processes span beyond the boundaries of single operation and cross over organizational boundaries.

Choreography and orchestration languages come to fill in that gap and to address the middle layer where atomic services are integrated. An orchestration language (such as Business Process Execution Language) specifies an executable process that involves message exchange with other systems, such that the orchestration designer controls the message exchange sequences. Choreography language specifies interactions and message exchanging through web services in a way to support interoperability and defines legal sequences and interactions among them. Choreography gives the flexibility to the user to select from many different interactions that comply with the language definitions.

The orchestration and the choreography distinctions are based on analogies: orchestration refers to the central control (by the conductor) of the behaviour of a distributed system (the orchestra consisting of many players), while choreography refers to a distributed system (the dancing team) without centralized control.

Technologies related to web services such as SOAP, UDDI and WSDL provide limited support to service discovery, service matching, choreography, orchestration and generally in the process of automated service composition. Furthermore the key actors, the experts of these processes, are not IT experts who nonetheless have to be involved in the design of business processes.

Natural language (NL) by itself is not appropriate for programming, but reference to NL is a key factor for the design of program instructions that are both machine-ready for processing and understandable to humans. For information retrieval and speech recognition systems NL reference is essential and aspect-oriented NL reference shows how powerful program organizations can be realized based on instructions and structures expressed in NL.
In a WS environment, services are propagated and identified through descriptions. To avoid ambiguity the descriptions have to be linked correctly into a coherent context map that allows, at the same time, to locate the required service. Semantic annotations such as domain ontologies or even simple control vocabularies related to a specific category are objects of coordination that can be communicated. The use of information retrieval methods can map the UDDI and WSDL descriptions to domain language objects. From a different angle this solution resembles the current successful application of NL processing in speech recognition systems where command languages are combined with NL processing interfaces to provide for robustness.

The objective of this chapter is to expand semantics used for web services by NL processing capabilities and enable users to write descriptions in their own language which can be transformed automatically into web service semantics. An integrated infrastructure, in a closed world, can give solutions to NL queries using web services.

An integrated environment, in a close world, can translate NL descriptions to specific queries, find relevant web services, orchestrate, choreograph them and return the desired results to the end user. Such a system hides complexity from the end user, for instance the complexity associated with XML, SOAP, UDDI, WSDL, BPEL and gives the opportunity to non IT people to benefit from web services. In the business area enterprises would lease each other’s server-side applications over the Internet and would plug into each other and do business over the wire, on the fly. Software companies co-operate on the fly and provide ad-hoc solutions that integrate their customers’ business processes.

In the following section we overview the background technologies (section 3) which are essential for the automatic composition of web services. In section 4 we analyse the methodology for a successful automatic web services composition. In section 5 we discuss the related work and conclude the chapter and discuss areas for further work (section 6) on the web services automation.

**BACKGROUND**

A lot of effort has been done for the standardization of the processes and the messages that enable interoperable integration between heterogeneous IT processes and systems. In next sections we focus on the existing standards about web services, semantics, business process execution language and natural language. In section 3.1 we discuss the service oriented architecture. Section 3.2 and 3.3 focuses on web services standards, and on standards about semantics and ontologies for web services respectively. In section 3.4 we discuss business process execution languages for web services and in section 3.5 we discuss natural language processing.

**Service Oriented Architecture (SOA)**

SOA (OASIS 2006) is an architectural style whose goal is to achieve loose coupling among interacting software agents. Service is a component which aims to fulfil a specific work for a service consumer. It’s an architectural paradigm that can be used to build infrastructures enabling those with needs (consumers) and those with capabilities (providers) to interact via services across disparate domains of technology and ownership.

Services are the core components of software applications which require a good orchestration in order to function properly. Several new trends in the computer industry rely upon SOA as the enabling foundation. Trends include: automation of Business Process Management (BPM), composite applications (applications that aggregate multiple services to function), or new architectural and design patterns generally referred to as Web 2.0.
Web Services Automation

Web Services (WS)

WS are frequently just web APIs that can be accessed over a network, and executed on a remote system hosting the requested services.

WS can be used to implement a service-oriented architecture. A major focus of WS is to make functional building blocks accessible over standard internet protocols that are independent from platforms and programming languages, to be self-contained, self-describing, modular applications that can be published, located, and invoked across the web. WS perform functions, which can be anything from simple requests to complicated business processes. An example of the WSs architecture is shown in Figure 1.

WS have showed the way to the formation of numerous standards and of industrial and academic consortia. WS are indeed useful for easier, faster integration, good in terms of return of investment, establishing friction free markets, and rapid value added assembly of services. WS represent an environment in which developers use components with well defined units of business logic and data access that can be assembled, at runtime, to enable a business process. However, some of the inherent problems of e-business like scalability and semantic interoperability are not solved by the service-oriented architecture provided by WSs. A WS offers a specific business function to applications provided by another application via standard internet protocols (XML, SOAP, WSDL, and UDDI). In the next sections we describe the SOAP, WSDL and UDDI protocols.

Simple Object Access Protocol (SOAP)

SOAP is an XML protocol for information exchange in a decentralized, distributed way. SOAP consists of three parts: the envelope which defines a framework for describing what is in a message and how to process it, a set of encoding rules for expressing instances of application-defined data types, and a convention for representing remote procedure calls and responses.

Web Services Description Language (WSDL)

The WSDL (WSDL) defines services as collections of network endpoints, or ports. A WSDL specification provides an XML format for documents for this purpose. WSDL handles set of ports, port types, operations and messages. A port is defined by associating a network address with a reusable binding, and a collection of ports define a service. Messages are abstract descriptions of the data being exchanged, and port types are abstract collections of supported operations. The concrete protocol and data format specifications for a particular port type constitute a reusable binding, where the operations and messages are then bound to a concrete network protocol and message format. WSDL describes the application programming interface to a WS.

Most of the times WSDL is used with SOAP and XML schema to publish WSs over the network. A client program connecting to a WS can read the WSDL to determine what functions are available on the server. Any special data types used are embedded in the WSDL file in the form of XML schema. The client can then use SOAP to actually call one of the functions listed in the WSDL.
Universal Description, Discovery and Integration (UDDI)

UDDI (OASIS UDDI, 2002) allows the discovery of potential business partners on the basis of the services they provide. Each business description in UDDI consists of a businessEntity element that describes a business by name, a key value, categorization, services offered (businessServices) and contact information for the business. Each businessService element contains descriptive information such as names and descriptions, and also classification information describing the purpose of the relevant WS. Using UDDI, a WS provider registers its advertisements along with keywords for categorization. The user retrieves advertisements out of the registry based on keyword search. So far, the UDDI search mechanism relied on predefined categorization through keywords, but more recently specifications to use OWL in UDDI are emerging as a uniform way to express business taxonomies.

Semantics and Ontologies for Web Services

Description of metadata is a key to semantic web and pillar technologies, which support description, are RDF and OWL. RDF is a data model for referring to objects (“resources”) and how they are related. Such a model is usually represented in XML syntax. RDF Schema is a vocabulary for describing properties and classes of RDF resources, with a semantics for generalization-hierarchies of such properties and classes. OWL adds extra vocabulary to describe properties, classes, and relations between classes.

In the area of the Semantic Web Services (SWS) there are various technologies and standards relevant to the semantic description of WSs like OWL-S (described in 3.3.1), SAWSDL (described in 3.3.2), Dublin core metadata (described in 3.3.3), WSDL-S (described in 3.3.4), SWSF (described in 3.3.5) and WSWO (described in 4.4.6).

Semantic Markup for Web Services (OWL-S)

OWL-S (OWL-S: OWL-based Web Service Ontology), formerly DAML-S, builds on top of OWL and allows for the description of a WS in terms of a Profile, which tells “what the service does/provides”, a process model, which tells “how the service works”, and a grounding, which tells “how to access the service” (Martin D & Paolucci M. & McIlraith S. & Burstein M., 2004). The service profile describes what is accomplished by the service, limitations on service applicability and quality of service, and requirements that the service requester must satisfy in order to use the service successfully. The process model gives details about the semantic content of requests, the conditions under which particular outcomes will occur, and, where necessary, the step by step processes leading to those outcomes. In the process model a service can be described as an atomic process that can be executed in a single step or a composite process that, similar to a workflow, can be decomposed in other processes based on control structures like ‘if-then-else’ and ‘repeat-while’. Finally, grounding descriptions supply information about the communication protocol and other transport information (such as port numbers) and the message formats and serialization methods used in contacting the service. The only currently specified grounding mechanism is based on WSDL 1.1 and has been extended to WSDL 2.0.

Semantic Annotations for WSDL

Semantic Annotations for WSDL and XML Schema (Semantic Annotations for WSDL and XML Schema, 2007) define how to add annotations to various parts of a WSDL document such as input and output message structures, interfaces and operations. The extension attributes defined in this specification fit within the WSDL 2.0, WSDL 1.1 and XML Schema extensibility frameworks.
For example, this specification defines a way to annotate WSDL interfaces and operations with categorization information that can be used to publish a WS in a registry. The annotations on schema types can be used during WS discovery and composition. In addition, SAWSDL defines an annotation mechanism for specifying the data mapping of XML Schema types to and from ontology; such mappings could be used during invocation, particularly when mediation is required. To accomplish semantic annotation, SAWSDL defines extension attributes that can be applied both to WSDL elements and to XML Schema elements. The semantic annotations reference a concept in ontology or a mapping document. SAWSDL is independent of the ontology language and this specification requires and enforces no particular ontology language. It is also independent of mapping languages and does not restrict the possible choices of such languages.

**Dublin Core Metadata**

The Dublin Core metadata element set (DCMI, 2009) is a standard (NISO Standard Z39.85-2001 and ISO standard 15836:2003) that provides a simple and standardized set of conventions for describing networked “resources” through a set of elements such as ‘Creator’, ‘Subject’, ‘Title’, ‘Description’, and so on. Dublin Core is widely used to describe digital materials such as video, sound, image, text and composite media like web pages. Implementations of Dublin Core are typically XML and RDF based.

**Web Service Semantics (WSDL-S)**

WSDL-S is a means to add semantics inline to WSDL. It is actually an extension to WSDL 2.0 but can be used for WSDL 1.1. According to WSDL-S inputs and outputs of WSDL operations are annotated with domain concepts while the operations themselves are annotated with preconditions and effects (post conditions). Also the service’s interface is annotated with category information which could be used while publishing services in registries such as UDDI. The semantic domain model used is external to these annotations and could be expressed in OWL or other ontology language of choice.

**Semantic Web Services Framework (SWSF)**

SWSF (Semantic Web Services Framework), initiated by the semantic WSs initiative (http://www.swsi.org/), includes the Semantic Web Services Language (SWSL) and the Semantic Web Services Ontology (SWSO). SWSL is a logic based language for specifying formal characterizations of WS concepts and descriptions of individual services. SWSO is ontology of service concepts defined using SWSL and incorporates a formal characterization (or the “axiomatic definition”) of these concepts in first-order logic.

**Web Services Modelling Ontology (WSMO)**

WSMO defines the modelling elements for describing several aspects of Semantic Web services. These elements are Ontologies, which provide the formal semantics to the information used by all other elements, Goals which specify objectives that a client might have when consulting a WS, WSs that represent the functional and behavioural aspects which must be semantically described in order to allow semi-automated use, and mediators that are used as connectors and they provide interoperability facilities among the other elements. It also defines the Web Service Modelling Language (WSML) which formalizes WSMO and aims to provide a rule-based language for the Semantic Web.
**BPEL for Web Services**


Each BPEL document specifies the behaviour of a business process. BPEL processes often invoke WS to perform functional tasks and can be either abstract or executable.

- Abstract processes describe inputs, outputs and act like application programming interfaces for processes. They also provide a description of what a process does.
- Executable processes do the “heavy weight lifting”—they contain all of the execution steps that represent a cohesive unit of work.

A process consists of activities connected by links. (A process sometimes only contains one activity, which is usually a container for more activities.) The path taken through the activities and their links is determined by many things, including the values of variables and the evaluation of expressions.

The starting points are called start activities. When a start activity is triggered, a new business process instance is created. From then on, the instance is identified by data called correlation sets. These data uniquely identify a process, but they may change over time. For example, the correlation set for a process may begin as a purchase order number retrieved from a customer order. Later, when an invoice is generated, the correlation set may be the invoice number.

BPEL is layered on top of other Web technologies such as WSDL, XML Schema, XPath, XSLT, and WS Addressing.

With the introduction of WS, terms such as “web services composition” and “web services flow” were used to describe the composition of WSs in a process flow. More recently, the terms orchestration and choreography have been used to describe this.

WS orchestration or choreography languages address the middle layer where atomic services (or operations) are integrated for more complex applications. In the following sections (3.4.1 and 3.4.2) we analyse choreography and orchestration of WS.

**Choreography**

Choreography deals with the message passing and the sequence between customers, suppliers, partners and anybody who is involved with the WSs. Choreography is typically associated with the public message exchanges that occur between multiple WSs, rather than a specific business process that is executed by a single party.

Web Services Choreography Language (Web Services Choreography Description Language) describes collaboration protocols of cooperating WS participants, in which services act as peers, and interactions may be long-lived and stable. WS choreography leverages the power of WSs to allow entities to create business processes that mirror today’s dynamic and ever-changing business needs. Organizations can expose their application software and resources as WSs so that others can dynamically find and use them in their business processes. WS-Choreography addresses the vision of true WS coordination and collaboration by providing practical models for dynamic, reusable and scalable process compositions and choreography, addressing technical completeness, correctness, and execution issues, enabling more
dynamic, semi-automated composed processes and enabling the incorporation of semantics.

Orchestration

Orchestration describes how WSs can interact with each other at the message level, including the business logic and execution order of the interactions. WS orchestration is the success story of SOA applications.

The SOA community has produced various competing specifications for WS orchestration. The Business Process Management Language for Web Services (BPML4WS) was the first mature specification available. BPEL4WS is a specification that models the behaviour of WS in a business process interaction. BPEL4WS is based on XML grammar for describing the control and the logic required to coordinate WSs in a workflow environment. BPEL4WS can be interpreted and executed by an orchestration engine, which is controlled by one of the participating parties. Industry has almost unanimously adopted the BPEL4WS orchestration description. One of the main drawbacks of BPEL4WS is the need for intensive human-tool interaction and in some cases the need for manual programming efforts.

Another promising standard proposal for WSs orchestration is WSCI. WSCI only describes the observable or visible behaviour between WS. In contrast to BPEL4WS WSCI, does not handle the definition of executable business processes. Furthermore, a single WSCI document only describes one partner’s participation in a message exchange. As the following illustrates, WSCI choreography would include a set of WSCI documents, one for each partner in the interaction. In WSCI, there is no single controlling process managing the interaction.

In order to cope with such challenges, the use of intelligent and distributed systems implementing peer-to-peer interactions has long been proposed.

Choreography vs. Orchestration

There is an important distinction between WS orchestration and choreography. Orchestration refers to an executable business process that may interact with both internal and external WS. For orchestration, the process is always controlled from the perspective of one of the business parties. Choreography is more collaborative in nature, in which each party involved in the process describes the part they play in the interaction. The term WSs orchestration is used here to describe the creation of business processes, either executable or collaborative, that utilize WSs.
Natural Language Processing

It is doubtful whether web standards together with WS orchestration and choreography standards suffice to unleash the full potential of software components that enable rapid development of versatile and highly adaptable applications. Natural language processing (NLP) (J Allen, 1995) can enhance these standards and give the ability to simple users to come closer together in the design of IT-based business processes.

Many NLP techniques, including stemming, part of speech tagging, compound recognition, de-compounding, chunking, word sense disambiguation and others, have been used in Information Retrieval (IR). Typical information retrieval systems search documents/texts in the large data store at either a single word level, or at a term level.

For every document a similarity score is assigned and information retrieval systems reveal subset of the original documents to the user. In most of the cases this subset has a relevancy score which is higher than a given or default score.

Semantic information is used in retrieval techniques to expand knowledge about documents and to foster performance. In one such system, NLP is used to match the semantic content of queries to that of the documents to be searched. Sentences or phrases are then translated to terms for indexing the documents to be searched.

NL can provide us with semantics to write requests in our language, and in a close world to compose and choreograph WSs in the way humans think.

WEB SERVICE AUTOMATION

WS are software systems designed to support interoperable machine-to-machine interaction over a network. They are the preferred standards-based way to realize SOA computing. While there has been a significant progress in this area, there are a number of factors that prevent the wide scale deployment of WS and creation of web processes. A perspective that has seen much interest from the research community is that of automated composition of WS. The most inherent problems for this approach concern discovering and orchestrating WS. The current solutions take a structural approach to describe WS using XML based definitions. The main problem that the community faced and tried to solve it by introduction the semantic WS, is that it’s not possible to explicitly define the purpose of the WSs as intended by the WS provider without semantics.

The ultimate goal is to realize WS compositions or web processes by leveraging the functionality of syntactic description of WS. Most of the composition standards build on top of WS description standards. Hence semantically describing a service could help in composing a process whose individual components are semantically described. Section 4.1 describes the web service automation in a seven layer architecture. Sections 4.2 to 4.6 analyse the layers of the architecture and in section 4.7 we give an example of the system.

Architecture

The objective is to develop a framework that enables experts in business processes rather than IT experts to define business processes. Experts will use just NL for the definition or description of processes. NL descriptions are transformed according to semantic standards into statements which will be enacted by computers. The transformation process, in turn, resorts to semantic web metadata. We focus primarily on coexistence of NL and semantic web standards in a layered architecture for automated enactment of WS.

With the use of semantic WS a service provider can enrich it’s applications with semantic descriptions. The OWL-S part of the description describes services in terms of semantics and is used in building the communication operations and an overall WS ontology.
The NL descriptions are matched with semantic representations of services. Input and output parameters of the services have to be considered in order to combine different services and generate a complete workflow. These representations help to identify the relevant services and to address correctly the information required by these services as well as the information produced by them. Matching process, in our point of view, is the key feature for a successful WS composition. Having a powerful service discovery application and semantic description of the corresponding WS, the process of choreography and orchestration become simple and effective.

In a fully automated WSs' environment the expert user must only provide a description in NL which can be used to develop the top layer of the architecture and which will have impact on the orchestration layer. Our objective is to make top layer’s semantics human-understandable. To achieve this goal we propose a layered architecture as follows:

1. **Natural Language Query**: Expert user posts a request as a description in natural language.
2. **Natural Language Parsing**: A system based on information retrieval and artificial intelligence applications is used to decompose/decrypt the user’s request to semantic and syntactic annotations using lexical resources such as WordNet, FrameNet and VerbNet.
3. **From Natural Language to Domain Language**: A domain language is based on a specific ontology for the domain. It is essential to have a specific ontology which will be used to translate NL to semantic description.
4. **Web Services Discovery**: A matching mechanism will trigger in order to find the appropriate services that decompose the user’s request. The service’s descriptions, semantic (OWL-S) and simple text (UDDI), will be translated to the domain language and with the translated request a simple matching algorithm will propose candidate WSs. In that phase the user can optionally evaluate that services which will participate in the WS orchestration.
5. **Choreography Construction**: With standards like WS-CDL the proposed WSs will be evaluated and the result is an automated composed process. WS-CDL describes interoperable collaborations between services. WS-CDL is like a contract containing the definitions of the constraints and the ordering conditions under which messages are exchanged. This describes the common and complementary observable behaviour of all the services involved from a global viewpoint.
6. **Orchestration Construction**: According to (Mendling & Hafner, 2005) “BPEL process definitions of all parties involved in a choreography can be derived from the global WS-CDL model”. Using such a system the orchestration can be generated using only the WS-CDL model.
7. **Enactment**: After a composite process is generated, the composite service is ready to be executed. Execution of a composite WS can be thought as a sequence of message passing according to the process model. The generated executable if it is composed in a BPEL4WS format can be executed in any BPEL execution engine.

**Natural Language Processing**

Having a non-IT expert as user it is essential to distinguish between the external (natural) language and internal (web service) language. NL used by the users to enhance accessibility in the sense that they can express what they want in a relatively easy manner. Internal language is used by the composition process generator, because the process generator requires more formal and
precise languages, for example, the logical programming languages.

Semantics’ web vision is to give access and knowledge to machines with semantic information which is normally used by humans. On the other hand, it will also revolutionize access to services by adding semantic information to create machine-readable service descriptions. NLP has advanced to the point where it can break the impasse and open up the possibilities of semantic web. NL systems can automatically create annotations from unstructured text. This provides the data that semantic web applications require.

NL search systems, such as Powerset (PowerSet), take advantage of semantic web markup and ontologies to augment their interpretation of underlying textual content. They can also expose SWS directly in response to NL queries using automatic techniques for semantic parsing. Such techniques have been successfully used in information extraction and question answering using a semantic parser.
A semantic parser can “decompose” the linguistic structure to extract meaning. An example of such a parser is XLE (Cahill & King & Maxwell, 2007). XLE consists of cutting-edge algorithms for parsing and generating Lexical Functional Grammars (LFGs). Another robust semantic parser is (Shi & Mihalcea, 2005) that uses a broad knowledge base created by interconnecting three major resources: FrameNet, VerbNet and PropBank. VerbNet lexicon provides the knowledge about the syntactic behaviour of the verbs. FrameNet thesaurus consists of examples annotated with semantic roles.

The linguistic interface layer is the key factor for a fully automated WSs environment. With NLP we ensure that the service proxy “understands” a phrase expressed in a descriptive way. “Understanding” means mapping an information item like a WS request into an appropriate request description. The linguistic layer contains thus features to map a service request or a service description (source language) into the target language generating automatically a request description that contains only terms of the domain ontology.

Domain ontology is a well defined description for the business process world that we focus. The domain ontology acts as a filter in user’s requests and in WS’s descriptions. Interpretation of NL to linguistic structure becomes feasible with the help of a semantic parser and public available sources like WordNet, FrameNet, VerbNet and PropBank (Giuglea & Moschitti, 2006). But what we finally need is to find services that will solve a specific problem. So the linguistic structure of user’s request must be translated to a common language with the WS’s descriptions. Domain ontology comes to fill in that gap. Using a filter over the domain ontology linguistic structure of the request and WS’s descriptions are translated to a common intermediate and domain specific “language”.

**Semantic Web Services Discovery**

WS selection is a crucial aspect of process composition. Semantic discovery is the process of discovering services based on the semantic metadata attached with the services. The use of ontologies allow richer description of activity requirements and more effective way of locating services to carry out the activities in the executable Web process.

WS described by semantics like OWL-S ontology become SWS, which enable programs to use the explicit semantics to automatically discover, invoke, compose and monitor the associated processes. Automatic discovery is accomplished by semantic matchmaking of a description of a required or sought process with available descriptions of SWS profiles. Because matchmaking is done using inference rather than one-to-one mappings, discovery can be performed based on skill sets rather than on device types.

Finding the right services is easier if service providers categorize services based on the domain with each registry maintaining only the WSs pertaining to that domain (Sivashanmugam & Miller & Sheth & Verma, 2005). If registries are specialized like this, it is possible to use domain specific ontologies. Environment specific information (in our case domain ontology) captures the characteristics of the environment where the universal (generic) processes are implemented. Universally applicable processes require information about the environment in which they are applied. For instance, a service for microarray discovery needs to know about the database where it can retrieve genes.

Furthermore, use of SWS enables working with multiple ontologies and, in particular, reasoning over unknown ontologies in order to identify recognized patterns. Semantic mappings between ontologies, such as term equivalence, enable link-
ing overlapping domains. However, the problem of translating ontologies is nontrivial, as more often than not direct translations are not possible.

**Automatic Choreography Construction**

Choreography is the abstract level that defines the transaction schema and interaction processes that must hold in order to execute a process through a WS.

A composite service query is generated based on the NLP of user’s request. The query profile includes the description of the composite service and the interface of the expected composite service, in which the output parameters, output constraints, input parameters, and their constraints have been extracted. The output constraint specifies the requirements on the outputs by the user.

Given that WSs can provide a suitable infrastructure for interactions and encapsulation of processes and skills and those OWL ontologies can provide the necessary machine-interpretable semantics, an approach that unifies these technologies is required.

WS’s ontology describes the interfaces of services and the relationships among them. An abstract service specifies names and types of input and output parameters with no property constraints. Like domain ontologies, a service inherits the properties of its parent service in WSs ontology (Zhang & Arpinar & Aleman-Meza, 2003). This requires capturing user’s goals and constraints, and matching them with the best possible composition of existing services. Therefore, inputs and outputs of the composite service should match the user supplied inputs, and expected outputs, respectively. Furthermore, the individual services placed earlier in the composition should supply appropriate outputs to the following services in an orchestrated way similar to an assembly line such as pipe-and-filter in a factory so they can accomplish the user’s goals. Finally, the composition should conform to the user specified constraints including time, cost, and user specified quality of composition properties.

Using WSCI or WS-CDL descriptions WSs can be formalized and check whether two or more services are compatible to interoperate or not (Brogi, A., Canal, C., Pimentel, E., Vallecillo, A, 2004). If services are incompatible then specification of adaptors that mediate between them can be automatically generated. In this context, compatibility can be described as the ability of two WS to work properly together, that all exchanged messages between them are understood by each other, and that their communication is deadlock-free.

**Automatic Orchestration Construction**

A complementary concept to service choreography is service orchestration. The orchestration level is concerned with the workflow-oriented execution and sequencing of atomic processes and it does not take into account the different types of conversation patterns required to invoke the implementing services that are associated with the abstract atomic processes of the workflow. An orchestration engine implements the application logic necessary to orchestrate atomic services, and provides a high-level interface for the composed process.

In order to execute a WS, its preconditions must be satisfied, possibly using information provided by other WSs. Moreover care has to be paid in avoiding the duplication of effects when composing services, which might be due to entailment relationships among different effects provided by services being composed.

Automatic composition is achieved by semantic matchmaking of atomic operations in a complex process orchestration to the skills described in discovered service ontologies (Jammes & Smit & Martinez-Lastra & Delamer, 2005). Enactment engines can thus execute complex processes by composing individual processes, enabling au-
tonomous manufacturing system orchestration. Semantics enable the chain of reasoning to traverse several ontologies in order to match post- and pre-conditions and compose services.

The main goal of transforming WS-CDL to BPEL is to allow the participants a rapid modelling and development process and generate relevant BPEL and WSDL documents which can then be used as a basis to implement the private (non-visible) business logic. According to (Mendling & Hafner, 2006) a dynamic formula can define the basic mapping rules from WS-CDL to BPEL. Using a recursive XSLT based approach we generate the BPEL processes by iterating through each role type of service to check its relevance.

Another approach based on time automata model which is verified and validated for correctness using formal model checking techniques have been proposed by (Diaz & Cambronero & Pardo & Valero & Cuartero, 2006). The key element is the WS-CDL document, in which each participant is represented as well as the time it enters into action. WS-CDL documents are translated into timed automata in a first step, and in a second step they intend to translate the timed automata into WS-BPEL documents.

**Enactment**

Automatic invocation is accomplished by interpreting the supported conversational semantics for discovered services. Consequently, the framework can select the appropriate choreography for service interactions. After a unique composite process, the services are ready to be executed. Execution of a WS workflow can be thought as a sequence of message passing according to the process model. The dataflow of the services is defined as the actions that the output data of a former executed service transfers to the input of a later executed atomic service.

The generated executable BPEL4WS process can be executed in any BPEL execution engine. The semantic web ‘vision’ is one of enhancing the representation of information on the web with the addition of well defined and machine readable semantics, thereby encouraging a greater degree of ‘intelligent’ automation in interactions with this complex and vast environment. WS composition involves ordering a set of atomic services in the correct order to satisfy a given goal.

**EXAMPLE**

We illustrate the proposed methodology with a generic procedure for knowledge discovery in the context of clinico-genomic trials. The experiment has been developed within the ACGT infrastructure. ACGT (Tsiknakis & Kafetzopoulos & Potamias & Marias & Analyti & Manganas, 2006) is an integrated project named “Advancing Clinico-Genomic Trials on Cancer: Open Grid Services for Improving Medical Knowledge Discovery”. ACGT is the provision of a unified technological infrastructure which will facilitate the seamless and secure access and analysis of multi-level clinico-genomic data enriched with high-performing knowledge discovery operations and services. The main advantage of ACGT infrastructure is that it supports a well defined master ontology (Mathias B. & Gabriele W. & Cristian C. & Holger S & Norbert G. & Martin D. et al., 2008) in the breast cancer domain.

Let’s assume that a physician wants to find ‘evidential’ correlations between patients’ gene groups and clinical profiles for a specific dataset. So a possible description from the physician would be “Give me correlations between gene groups and Metastasis for the public dataset Veer”.

The first step of the process is the linguistic analysis. The physician’s description will be enriched with synonyms. Semantics and syntactic decomposition of the phrase will also come up. At the next step a filtering according to the domain ontology, which in our case is the master
ontology of ACGT, will start. The same holds for the descriptions (OWL-S and UDDI) of the WSs in the ACGT infrastructure. Having a common filtering of the NL description and the service's descriptions we create a common language (or simply a vocabulary) for services and people.

Then a matching procedure reveals the candidate services. The semantic WSs within ACGT that reveals in response to the physicians request are three. The mediator service, the association rules service and the discretized k-means service. The mediator is a service which takes as input a query and gives as output a zip file with merged data of clinical and genomics for each patient. Mediator has been proposed by the WS discovery service because of the keywords “public” and “dataset”. Discretized K-Means is a special implementation of the known algorithm k-means within ACGT. The keyword that revealed this service is “group”. This keyword appears in the semantic description of discretized k-means service. Association rules mining service implements a machine learning algorithm called apriory on XML data and gives relations between attributes. Detailed description of the knowledge discovery services is beyond the scope of this chapter and can be found in (Potamias & Koumakis & Kanterakis & Sfakianakis & Analyti & Moustakis et al, 2007). Association rules mining service has been proposed by the WS discovery service because of the keyword correlation which is synonym to association and appears in the description and in the master ontology.

Having these three services the system will try to combine (orchestrate) and generate a complex service which will give the desired output. Based on the WS-CDLs of the services we can see that mediator service need as input a string which is the query and give as output a zip of clinical and genomic data. The discretized k-means need as input a zip file with the clinical and genomic data and give as output a zip file with clinical and clusters of genomic data. Association rules mining service need as input an xml file with clinical and genomic data and give as output a file with associations between clinical and genomics. As we can see the output of mediator can be input at k-means algorithm but the output of k-means is incompatible with the input of association rules mining service. In this case the system will try to automatically find services that act as intermediates (or filters) for data type conversion. This search is based on the data types of input and output services. The search mechanism finds a new

Figure 5. workflow editor and enactment environment (WEEE) of ACGT
service which is called Kmeans2ARMfilter and is a simple transformation of the output zip files of k means service to XML which is the desired input for association rules mining. Have also this service the system can orchestrate the available services in order to generate a workflow.

In the next step the system propose a sequential workflow where the mediator service is the first one to invoke. The input of this service (the query) must be specified by the user. Then the output of mediator goes to discretized k-means service which clusters the genomic data. After the clustering data goes to Kmeans2ARMfilter service which transforms the output of discretized k-means to input for association rules mining service. Finally the association rules mining service generate associations between clinical and genomics data and stores them to the file system. Figure 5 shows the workflow which has been designed with the workflow editor and enactment environment (WEEE) of ACGT.

RELATED WORK

Automatic composition of semantic WSs is one of the most interesting and challenging problem in the field of semantic web. Current trends in industry and in research increase the number and the heterogeneity of WSs making automatic composition harder but also essential in large scale applications.

Benatallah at al (Benatallah & Dumas & Sheng & Ng, 2002) proposed a platform called SELF-SERV which implements WSs composition using state charts, where the resulting services can be executed in a decentralized way within a dynamic environment. This execution model allows services participating in a composition, to collaborate in a peer-to-peer fashion in order to ensure that the control and data flow dependencies expressed by the schema of a composite service are respected.

METEOR-S (Sivashanmugam, Miller, Sheth, & Verma, 2005) is a project which focuses on applying semantics in annotation, quality of service, discovery, composition and execution of semantic WSs. The project focuses on adding semantics to WSDL and UDDI o BPEL4WS, and proposes a semi-automatic approach for annotating WSs described using WSDL. The METEOR-S WS annotation framework leverages schema matching techniques with a Naïve Bayesian Classifier, for semi-automatic annotation. The composition process involves a functional perspective and an operational perspective. The functional perspective involves WS discovery, addressing semantic heterogeneity handling. The operational perspective takes form of the research on quality of service specification for WSs and processes.

SWORD (Ponnekanti & Fox, 2002) is a set of tools for the composition of a class of WSs including ‘information-providing’ services. SWORD manages service composition as a sequence of operations the compute data. In SWORD, a service is represented by a rule that expresses certain inputs and is capable of producing particular outputs. A rule-based expert system is then used to automatically determine whether a desired composite service can be realized using existing services. If so, this derivation is used to construct a plan that when executed instantiates the composite service. SWORD focuses on the composition of services based on descriptions but it does not address the problem of mismatching inputs/outputs of WSs.

CONCLUSION AND FUTURE WORK

WSs represent the second generation of Internet tools. Standards are undoubtedly important for this technology. In most of the cases, applications emerge from a complex and dynamic composition of a number of WSs. Almost all business processes span beyond the boundaries of single
Web Services Automation

operation and cross over organizational boundaries. Choreography and orchestration languages describe two aspects of emerging standards for creating business processes from multiple WSs. In this chapter we expand semantics used for WSs by NLP capabilities and enable users to write descriptions in their own language which can be transformed automatically into WS semantics. Robust text mining methods that map descriptions into a suitable domain ontology ensure that the descriptions are machine-process useable.

We have shown that it is feasible to create an integrated infrastructure for automatic composition of WSs in a closed world is which will give solutions to NL requests. To serve these requests, we expand semantics for WSs by NLP capabilities. Standards like OWL-S and WS-CDL on semantic WSs give us the ability and flexibility to select appropriate services and orchestrate them in order to answer NL questions automatically. The system is based in a “closed” world which means that the services and the domain are specific. Furthermore domain ontology helps us at the interpretation of requests and at searching over the services in order to have effective and accurate results. In our example we selected the ACGT environment because it has a specific domain (breast cancer) with a well defined domain ontology (ACGT master ontology) and support semantic WSs.

The proposed architecture can be expanded outside the boundaries of a specific domain using statistical and information retrieval techniques for automatic domain ontology generators. Furthermore an ontology evaluator and evolution system based on text analysis algorithms using precision/recall and F-measure methods could support the maintenance of the ontology. Such a method take as input documents selected by the user and update the ontology in real time. The input documents can be also the visited web pages of the user. Having this in mind we can pass to automatic composition of complex WSs from the domain specific world to user specific world.

REFERENCES


**KEY TERMS AND DEFINITIONS**

- **NLP:** Natural Language Processing
- **SWS:** Semantic Web Services
- **OWL-S:** Web Ontology Language(OWL) based framework of the Semantic Web
- **WS-CDL:** Web Services Choreography Description Language
- **BPEL4WS:** Business Process Execution Language for Web Services
# Appendix: Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>B2B</td>
<td>Business to Business</td>
</tr>
<tr>
<td>BPEL</td>
<td>Business Process Executable Language</td>
</tr>
<tr>
<td>BPEL4WS</td>
<td>Business Process Execution Language for Web Services</td>
</tr>
<tr>
<td>DAML-S</td>
<td>DARPA agent markup language for services</td>
</tr>
<tr>
<td>HTML</td>
<td>HyperText Markup Language</td>
</tr>
<tr>
<td>IR</td>
<td>Information Retrieval</td>
</tr>
<tr>
<td>ISO</td>
<td>International Organization for Standardization</td>
</tr>
<tr>
<td>IT</td>
<td>Information Technology</td>
</tr>
<tr>
<td>LFG</td>
<td>Lexical Functional Grammars</td>
</tr>
<tr>
<td>NL</td>
<td>Natural Language</td>
</tr>
<tr>
<td>NLP</td>
<td>Natural Language Processing</td>
</tr>
<tr>
<td>OWL</td>
<td>Web Ontology Language</td>
</tr>
<tr>
<td>OWL-S</td>
<td>Web Ontology Language(OWL) based framework of the Semantic Web</td>
</tr>
<tr>
<td>RDF</td>
<td>Resource Description Framework</td>
</tr>
<tr>
<td>SAWSDL</td>
<td>Semantic Annotation of WSDL</td>
</tr>
<tr>
<td>SOA</td>
<td>Service Oriented Architecture</td>
</tr>
<tr>
<td>SOAP</td>
<td>Simple Object Access Protocol</td>
</tr>
<tr>
<td>SWS</td>
<td>Semantic Web Services</td>
</tr>
<tr>
<td>SWSF</td>
<td>Semantic Web Services Framework</td>
</tr>
<tr>
<td>SWSL</td>
<td>Semantic Web Services Language</td>
</tr>
<tr>
<td>SWSO</td>
<td>Semantic Web Services Ontology</td>
</tr>
<tr>
<td>UDDI</td>
<td>Universal Description, Discovery and Integration</td>
</tr>
<tr>
<td>W3C</td>
<td>World Wide Web Consortium</td>
</tr>
<tr>
<td>WS</td>
<td>Web Service</td>
</tr>
<tr>
<td>WS-CDL</td>
<td>Web Services Choreography Description Language</td>
</tr>
<tr>
<td>WSCI</td>
<td>Web Service Choreography</td>
</tr>
<tr>
<td>WSDL</td>
<td>Web Service Description Language</td>
</tr>
<tr>
<td>WSDL-S</td>
<td>Semantic Web Service Description Language</td>
</tr>
<tr>
<td>WSML</td>
<td>Web Service Modelling Language</td>
</tr>
<tr>
<td>WSMO</td>
<td>Web Services Modelling Ontology</td>
</tr>
<tr>
<td>WSRF</td>
<td>Web Services Resource Framework</td>
</tr>
<tr>
<td>XML</td>
<td>Extensible Markup Language</td>
</tr>
<tr>
<td>Xpath</td>
<td>XML Path Language</td>
</tr>
<tr>
<td>XSLT</td>
<td>Extensible Stylesheet Language Transformations</td>
</tr>
</tbody>
</table>