***Paper** send to the Organizing Committee of the "W3C Workshop on Frameworks for Semantics in Web Services", June 9-10, 2005, Digital Enterprise Research Institute (DERI), Innsbruck, Austria*

Intelligent Framework for Generating Open (Adaptable) Development Platforms for Web-Service Enabled Applications Using Semantic Web Technologies, Distributed Decision Support Units and Multi-Agent-Systems

H.-Joachim Nern*, G. Agre*, T. Atanasova*, A. Micsik*, L. Kovacs*, T. Westkaemper*, J. Saarela*, Z. Marinova*, A. Kyriakov*

* Developer partners of the INFRAWEBS EU project, FP6-IST-511723, IST Strategic Objective: 2.3.2.3 Open development Platforms for software and services

Project acronym: **INFRAWEBS**

Stats & Facts:

Start date:	2004-Aug-01
End date:	2007-Jan-31
Participants:	University of Applied Sciences, Bochum, Germany; University of Innsbruck, Austria; Bulgarian Academy of Science, Institute of Information Technology, Bulgaria; MTA SZTAKI, Hungary; National Technical University of Athens, Greece; Profium SA, Finland; Sirma SAI, Bulgaria; FUTUREtec-GmbH, Germany; Atos Origin, Spain; Best-HP, Italy; big7.net GmbH & Aspasia Knowledge Systems, Germany
Costs:	3.1 million EURO
Project Site:	**http://www.infrawebs.org**

Summary

The paper gives an overview about the ongoing INFRAWEBS project and describes the main modules and software components embedded in an application oriented realisation framework. With regard to the SWS technology the inherently given message and position of this paper is to leave the conceptional level towards practical and reasonable applicable software tools and components.

Main Idea & Project Objective

The primary project objective is to develop an ICT framework consisting of several specific software tools (INFRAWEBS Unit), which enables software and service providers to generate and establish open and extensible development platforms for Semantic Web Service based applications /1/. This software tool set facilitates the establishment of virtual development platforms as well as interoperable middleware designed for a semantic and ontology-based handling of Semantic Web Services oriented on given conceptional WSMO specifications.
Generated in this way, the open platform (Fig. 1) consists of coupled and linked INFRAWEBS units, whereby each unit provides tools and system components to analyse, design and maintain WEB-Services realised as Semantic-Web-Services within the whole life cycle.

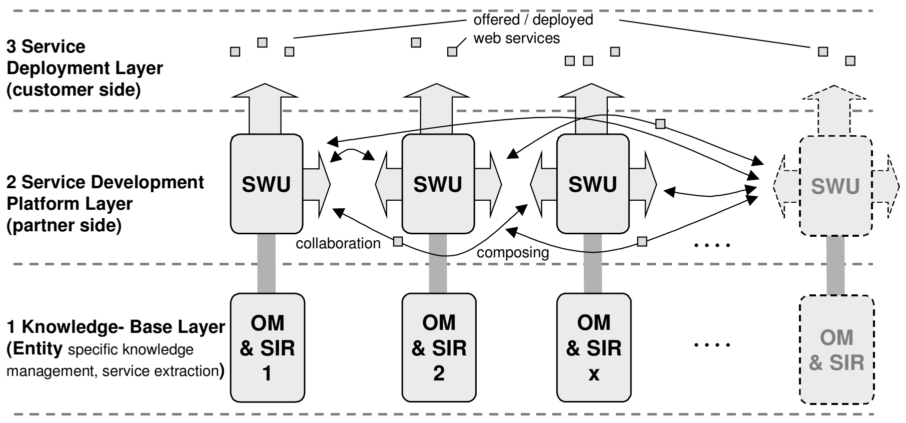
offered / deployed web services
3 Service Deployment Layer (customer side)
2 Service Development Platform Layer (partner side)
SWU
SWU
SWU
SWU
collaboration
composing
....
1 Knowledge- Base Layer (Entity specific knowledge management, service extraction)
OM & SIR 1
OM & SIR 2
OM & SIR x
....
OM & SIR

Fig. 1: Open and extensible development platform for the design, deployment and maintenance of Semantic Web Services as a net of **coupled INFRAWEBS** units.

As illustrated in **Fig.1** the overall design is structured in three main layers:
1) a knowledge management layer for handling service related knowledge artefacts realised as an organisational memory coupled to semantic information routing components (OM&SIR),
2) a service development layer for creating and maintaining Semantic Web Services embedded in a semantic based interoperable middleware, consisting of Semantic Web Service Designer & Composer, Distributed Semantic Web Service Registries, and an agent based discovery module (Semantic Web Service Unit -SWU)
3) a service deployment layer for the execution and monitoring of Semantic Web Services exploiting closed loop feedback information (Quality of Service brokering) provided for distributed decision support issues.
The software tool-set building components map the specific modules of the INFRAWEBS framework resp. unit. In the following this modules and their interdependencies are described:

Fuzzy Concept Based Organizational Memory - OM

As a repository for semi-structured knowledge artefacts a Fuzzy Set based organisational memory (FCM-OM) will be realised for knowledge management issues in the knowledge management layer /2/. The OM management module is endowed with tools for knowledge acquisition and knowledge representation. This module is responsible for the collection, organisation, refinement, and distribution of knowledge objects handled and managed by the service providers. The current specification and realisation of the INFRAWEBS OM is based on the novel results of research activities in the area of Fuzzy Set theory:
Considering the ambiguity, imprecision of concepts (electronic knowledge objects of an entity, including the objects handled and received in Internet related environments) an useful approach is the adaptation and application of FCM (Fuzzy Conceptional Matching) methods /3/. Within this approach a “concept” is defined (and represented) by a sequence (a set) of weighted keywords /3/. Ambiguity in this concepts is defined by a set of imprecise concepts. Whereas each imprecise concept is defined as a set of fuzzy concepts (using methods of implicit semantics), which is related to “a set of imprecise terms representing the context” /3/.
Involving and considering also formal semantics, this imprecise terms (words) are “translated” into precise terms (words) formalised as an ontology /3/.
Within the INFRAWEBS project two streams are focused in realising this system component:
- one component of the OM (FCM-OM) is designed following the rules of implicit & soft semantics (using statistical based AI methods like clustering and classification)
- a second component (O-OM) reflects the Knowledge handling based on methods related to formal semantics (Ontologies) – hard semantics /3/.

Semantic Information Routing - SIR

A further module within the first platform layer is the Semantic Information Router (SIR) /4/ which is coupled to the OM.

SOAP interface to OM

The interface between the Semantic Information Router (SIR) and the OM is based on the SOAP protocol. The interface is used to provide WSDL based service descriptions to the OM component and to query for content item references that are related to given service description references.

SPARQL interface for querying service descriptions from SIR

The query interface of the SIR component is based on the **SPARQL protocol**. SPARQL is a RDF metadata query language designed by the W3C DAWG working group with protocol bindings defined for JDBC, HTTP and other protocol stacks. Profium Ltd. is one of the companies present in the W3C DAWG working group. Technically the SPARQL interface is a metadata query language with syntax close to SQL which offers querying metadata as table and graph result sets. The protocol stack most suitable for J2EE environments is a JDBC type 4 (Java Database Connectivity) compliant driver. Profium Ltd. is also involved in the implementation of a JDBC based protocol binding for SPARQL named SPARQL4j (http://sourceforge.net/projects/sparql4j).

Service registration in SIR

Registration of non-semantic atomic services is accomplished via a web-based GUI interface. The interface is mainly used to input WSDL and BPEL4WS based service descriptions and enter additional related non-functional properties. Additionally a UDDI registry interface is provided for the SIR component. This will be established via SOAP as a UDDI Subscription Listener.

SWS Designer & Composer - SWU

The main module within the second – the service development – layer is the Semantic Web Unit /5, 6, 7/. Within this unit the modules Designer and Composer are responsible for decision supported creation of Semantic Web Services using the **Case-Based Reasoning** approach. The Designer is a tool for semiautomatic conversion of non-semantic Web services to Semantic Web Services, whereas the Composer enables the semiautomatic creation of new Semantic Web Services via the composition of existing Semantic Web Services.

Designer
The Designer consists of several sub-modules responsible for WSMO compliant creation of main elements of a INFRAWEBS specific Semantic Web Service, consisting of WSDL, BPEL4WS and UDDI files.
As a graphical user-friendly tool the **Capability Editor** facilitates construction and editing of complex WSML-based logical expressions used for representing service capabilities. The **BPEL4WS-based editor** serves for creating WSMO-based service choreography and orchestration. The **Grounding Editor** provides facilities for semiautomatic creating of WSMO-based grounding. A basic feature of the Designer is the use of **Design Service Templates** (DST), representing graphical models of capability and functionality (or their parts) of Semantic Web Services, which have been designed by the user in the past.
Composer
The Composer presents an interactive approach for composition of WSMO compliant Semantic Web Services, whereby its operation is based on **Service Composition Templates** (SCT). This SCT represents graphical models of the service composition as a control and a data flow between several semantic sub-services given by incomplete description of their capabilities. On the (Service) provider side the Composer enables the creation of a new composed "static" Semantic Web Service by discovering appropriate Semantic Services matching the required capabilities. Selecting of such services is implemented as an interactive system-driven semiautomatic process.
The Designer and Composer are implemented in the J2SDK 1.4.2 Runtime environment. Eclipse RCP (Rich Client Platform) is used for developing the basic platform components, plug-in infrastructure and graphical user interface components, whereas Eclipse GEF (Graphical Environment Framework) is the basis for implementing the graphical editors. Access to WSMO-based repositories (ontologies, Semantic Web Services, etc.) is realized via the **WSMO API**.

WSMO API - WSMO4J

For ensuring compatibility and interoperability within and between the INFRAWEBS framework modules the WSMO API (WSMO4J) is applied. The WSMO4J is an open-source project (distributed under a LGPL licence) with two parts: a) **WSMO API** - application programming interfaces for WSMO, which allow basic manipulation of WSMO descriptions, e.g. creation, exploration, storage, retrieval, parsing, and serialization and b) WSMO4J - a **reference implementation** of the WSMO API, including a WSML parser.
One of the major advantages of using the WSMO API in INFRAWEBS is to assure the compatibility and interoperability between the SWS Designer and Composer modules, and the repository component. For example, the distributed SWS registry uses WSMO4J in the process of transforming WSMO element descriptions into RDF triples stored into an RDF triple repository for efficient query and management. Using WSMO4J enables easy integration and interoperability within the framework as well as with the WSMO Studio, thus some of the components can be realized as extensions (plug-ins) for the WSMO Studio.

Agent Based SWS Discovery - SUA

The SUA (service and user agent) /8/ unit is a basic layer of the INFRAWEBS software environment (Fig. 2). The Communicator is responsible for building up the communication channels between users and various other modules. The User circle denotes the application acting on behalf of the user: for example a GUI interface or an intelligent agent. The **User circle** and the **SWS Composer** need to discover and select existing web services with specified capabilities. The Discovery Agent supports this task, while the Service Agent supports the execution of the selected web service in cooperation with the SWS Executor module.

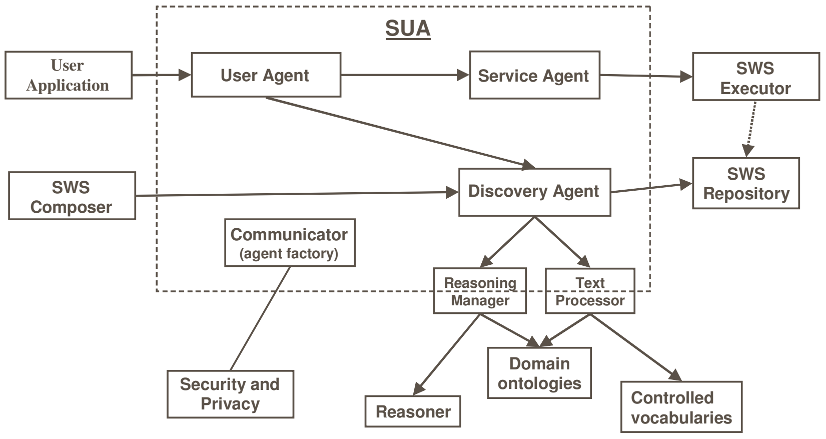

Fig. 2: Internal Architecture of Middleware Layer and its Dependencies with Other Modules

The **discovery process** is planned as a hybrid approach combining text processing and reasoning. Therefore, this process needs other external information and tools as well: domain ontologies contain the background information about service domains and organizational memories gathered from the INFRAWEBS framework and transformed into Semantic Web compatible format.
The SUA unit within INFRAWEBS acts as a **middleware layer** for user applications, connecting these applications to the functionality available in the form of Semantic Web Services. An important decision is whether to hide the semantic approach and accompanying logic-based framework of SWS from the user applications or not. An present investigation is to clarify: What level to choose for communication? On the level of plain web services, simple XML data structures are exchanged, which are easy to generate and process, but lack the possibilities of the semantic approach. On the level of semantic web services, facts and rules are exchanged in the form of logical expressions. As the latter case puts extra requirements for the user application, it is decided to find a middle course between the two solutions. Communication is kept on the semantic level, but its form is either hidden or aided with special functionalities and automatic translations.
Apart to SWS discovery the simplest way is based on **keyword matching** within the descriptive metadata of web services (similar to the UDDI approach). The most complex way is to logically prove that the web service is able to fulfill the given goal. Currently, response time of discovery process is a significant tradeoff for these approaches.
The **keyword-based method** is improved if the goals and capabilities are used for keyword generation instead of metadata. Goals and capabilities are described using ontology terms, so more homogeneous and precise keywords can be extracted this way. The **logic-based methods** differ in aspects of goals and capabilities considered. It is simpler and faster to match only by postconditions, but then matching services might not be executable because of missing or unacceptable information (input).
The project approach is to **split the discovery into two phases**. In the first phase, the keywords of capabilities are used to filter the possible web services. Then, logical matching is applied only for the filtered services.
The list of matching web services is returned to the user application enriched with descriptive and qualifying metadata.
A Web service is the access point to the real service: its quality determines the quality of access, but might be independent of the quality of service (e.g. automatic translation or flight reservation). Therefore, an **iterative selection process** (similarly to iterative query refinement in information retrieval) guides the selection of the best service in INFRAWEBS. This is based on a simple **two phase workflow agreement** (or business logic) between the web services and the middleware layer.
The SUA component of INFRAWEBS serves as a generic middleware for deployment of Semantic Web Services. It provides support for the usual steps of **goal construction**, **discovery**, **selection** and **execution** of Semantic Web Services. This support is achieved through a neutral interface, which hides the complexities of Semantic Web Services, therefore applications can be easily adapted to it, and also it can be equivalently used in variants of Semantic Web Services, such as WSMO and OWL-S. SUA also features an iterative selection refinement process for finding not only the suitable web services, but also the best service offers for users' goals.

SWS Executor & QoS-Aspects

SWS Executor

The SWS executor is split up into three main components, namely, the Communication Manager, Choreography Engine and the Invoker /9/. At present it is defined that the executor should mainly interact with the distributed registry and the SUA components. The SUA sends a package request to the Communication Manager. The latter component is responsible to handle communication between the external entities and the components inside the Executor. From the SUA it receives a package request which contains information about the particular package the user wants to execute. This request is addressed to the repository in order to fetch the already validated package. This validated package is then forwarded to the Choreography engine for execution. It is thought that the choreographies will be state based and that each state grounds to particular WSDL operation . The service request information is forwarded to the Invoker. The Invoker will be based on a standard API such as the Web Service Invocation Framework (WSIF). Once an operation is completed, a service result is forwarded to the choreography engine. This result will determine the next operation to do. It may be either some data which will be fed in to the next state or a failure to execute some service. The Choreography Engine will send execution information at each step to the Communication Manager. The latter will keep such information and send it back to the SUA once execution is terminated. This information will include errors, traces and results of the execution.

QoS-Aspects

Within the QoS broker development several aspects are addressed in order to make effective use of QoS in the INFRAWEBS architecture. An agreed, sound and complete definition of the various metrics is given to enable uniform measurement and comparability. Besides a sufficiently precise description, this definition includes the measurer itself since it is not always possible to make meaningful measurements for all quality aspects from within a single entity. Consider the example of the execution time of a service or package of services, which

includes not only the time spent processing but also the time that is spent for transmitting data across the network versus the process time. Given this definition for atomic services one a set of functions is defined that aggregate QoS metrics of atomic services within a single package and return a vector of metrics that is a sound representative for the package.
It is distinguished between a composite service and a (possibly) atomic service used by a composite one. For every composite service, an invocation to another service as atomic is considered. Thus, from a composite description perspective, whether an invocation calls a composite or a single operation of a web service is hided. This simplistic view enables to define atomic metrics over single invocation and furthermore define a mathematical model for the calculation of QoS metrics for a composite service. However, it must be noted that in the general case, a precise value for some quality criterion for a composite service cannot be determined prior execution of the composite service. This is due to non-deterministic execution flow for certain control constructs such as loops and forks.

Use Case

The use case part of the project is designed by the industrial partners. This use case is defined in the area of e-Tourism, particularly, it addresses a frequent flyer programme called STREAM Flows! System (SFS for short). This system will be available online and it can be accessed both from customers and service providers. The industrial partner specifies the required WSMO compliant semantic descriptions - by specifying the Ontologies, Web Services, Goals and Mediators.
Customers will be able to group and specify the services needed in terms of packages. A package is a collection of services which are to be purchased by the user. It may be either pre-defined by the SFS Administrator or customized by the customer. Service providers can publish their services with SFS in order to make them available for customers. It is up to the SFS to update the stored packages once a service has changed. The project use case shall take the following path:

* The Service Providers will have published some services: a flight, a hotel reservation, and a car rental service
* The customer searches for a package of desired services
* The SFS program access to the catalogue and search for the best services that fulfil the customer requirements

References

WSMO4J - http://wsmo4J.sourceforge.org
WSMO Studio - http://www.wsmostudio.org/
SPARQL4j - http://sourceforge.net/projects/sparql4j
Use Case - http://www.wsmo.org/TR/d3/d3.6/v0.1/20050408/

/1/ H Joachim Nern, G. Agre, T. Atanasova, J. Saarela. 2004, “System Framework for Generating Open Development Platforms for Web-Service Applications Using Semantic Web Technologies, Distributed Decision Support Units and Multi-Agent-Systems - INFRAWEBS II. WSEAS TRANS. on INFORMATION SCIENCE and APPLICATIONS, 1, Vol. 1, 286-291

/2/ H Joachim Nern, A Dziech, E Tacheva, “Fuzzy Concept Sets (FCS) applied to semantic organizational memories within the Semantic Web Service designing and composing cycle”, 1st Workshop for "Semantic Web Applications" at the EUROMEDIA 2005, April 2005, IRIT, Université Paul Sabatier, Toulouse, France

/3/ Lotfi A. Zadeh. Toward a perception-based theory of probabilistic reasoning with imprecise probabilities. In Journal of Statistical Planning and Inference 105 (2002) 233-264.

/4/ T Westkaemper, J Saarela, H Joachim Nern, “Semantic Information routing as a pre-process for Semantic Web Service generation - SIR & OM”, 1st Workshop for "Semantic Web Applications" at the EUROMEDIA 2005, April 2005, IRIT, Université Paul Sabatier, Toulouse, France

/5/ Tatiana Atanasova, Gennady Agre, H Joachim Nern, “INFRAWEBS Semantic Web Unit for design and composition of Semantic Web Services INFRAWEBS approach”, 1st Workshop for "Semantic Web Applications" at the EUROMEDIA 2005, April 2005, IRIT, Université Paul Sabatier, Toulouse, France

/6/ H Joachim Nern, A Dziech, T Atanasova, Applying Clustering and Classification Methods to distributed Decision Making in Semantic Web Services Maintaining and Designing Cycles, EUROMEDIA 2005, Workshop for "Semantic Web Applications", April 11-13, 2005, IRIT, Université Paul Sabatier, Toulouse, France

/7/ Gennady Agre, Tatiana Atanasova, H Joachim Nern, “Case Based Designer and Composer”, 1st Workshop for "Semantic Web Applications" at the EUROMEDIA 2005, April 2005, IRIT, Université Paul Sabatier, Toulouse, France

/8/ L Kovacs, A Micsik, “The SUA-Architecture within the Semantic Web Service Discovery and selection process”, 1st Workshop for "Semantic Web Applications" at the EUROMEDIA 2005, April 2005, IRIT, Université Paul Sabatier, Toulouse, France

/9/ A Polleres, J Scicluna, “Semantic Web Execution for WSMO based choreographies”, 1st Workshop for "Semantic Web Applications" at the EUROMEDIA 2005, April 2005, IRIT, Université Paul Sabatier, Toulouse, France