FP6-511723

INFRAWEBS

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

Specific Targeted Research Project
Priority 2 “Information Society Technologies”

Deliverable D1.3-4.10
Publishable Final Activity Report

Authors
Gennady Agre
Zlatina Marinova
Andras Micsik
Tomas Pariente Lobo
Janne Saarela
James Scicluna
Ettore Riceputi
Elpida Tzafestas
Alejandro Lopez

Editors
Joachim Nern
Tanja Atanasova
Caterina Rehm-Berbenni
Jose Estevez
Friedbert Pautzke
Steffen Saalmann

Due date of deliverable: 31/01/2007
Actual submission date: 22/05/2007

Start date of the project: 01/08/2004
Duration: 30 Months

Prof. Dr.-Ing. Friedbert Pautzke
University of Applied Sciences Bochum
This is the final report of the IST project INFRAWEBS (FP6 – 511723). It comprises a publishable overview about the INFRAWEBS project and describes the main project objectives and achievements as well as the structures embedded in the application oriented Software framework.

The INFRAWEBS Project was conceived to realise a platform for designing, maintaining, and executing of Semantic Web Services (SWS). The designing process is based on existing Web services and oriented on the WSMO Framework - a European initiative for Semantic Web Services, which provides a formal ontology and language for describing the various aspects related to SWS. Conceptually, the INFRAWEBS Integrated Framework (IIF) consists of coupled and linked INFRAWEBS semantic Web units (SWU), whereby each unit provides tools and components for analyzing, designing, and maintaining WSMO based SWS within the whole life cycle.

The most important result is, that the INFRAWEBS Framework is one of the first consistent frameworks for SWSs that covers the whole life-cycle of SWS, from the design to the discovery, composition, execution, and monitoring of services.
Abstract
This is the final report of the IST project INFRAWEBS (FP6 – 511723). It comprises a publishable overview about the INFRAWEBS project and describes the main project objectives and achievements as well as the structures embedded in the application oriented Software framework.

The INFRAWEBS Project was conceived to realise a platform for designing, maintaining, and executing of Semantic Web Services (SWS). The designing process is based on existing Web services and oriented on the WSMO Framework - a European initiative for Semantic Web Services, which provides a formal ontology and language for describing the various aspects related to SWS. Conceptually, the INFRAWEBS Integrated Framework (IIF) consists of coupled and linked INFRAWEBS semantic Web units (SWU), whereby each unit provides tools and components for analyzing, designing, and maintaining WSMO based SWS within the whole life cycle.

The most important result is, that the INFRAWEBS Framework is one of the first consistent frameworks for SWSs that covers the whole life-cycle of SWS, from the design to the discovery, composition, execution, and monitoring of services.

Version log/Date
Change
Author
<table>
<thead>
<tr>
<th>Version</th>
<th>Action</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>V01/29/11/06</td>
<td>Basic Frame</td>
<td>Steffen Saalmann</td>
</tr>
<tr>
<td>V02/10/01/07</td>
<td>Contributions added</td>
<td>Steffen Saalmann</td>
</tr>
<tr>
<td>V03/21/02/07</td>
<td>Revise</td>
<td>Steffen Saalmann</td>
</tr>
<tr>
<td>V04/21/02/07</td>
<td>Revise</td>
<td>Friedbert Pautzke</td>
</tr>
<tr>
<td>V05/21/02/07</td>
<td>Included comments from Big&amp;Aspa, ATOS, and Ftec</td>
<td>Friedbert Pautzke</td>
</tr>
<tr>
<td>V06/19/03/07</td>
<td>Added new chapter 4: INFRAWEBS FRAMEWORK MODULES</td>
<td>Joachim Nern</td>
</tr>
<tr>
<td>V06/20/03/07</td>
<td>Rework of chapter 4</td>
<td>Tanja Atanasova</td>
</tr>
<tr>
<td>V06/26/03/07</td>
<td>Revise</td>
<td>Tomás Pariente</td>
</tr>
<tr>
<td>V07/12/04/07</td>
<td>Revise</td>
<td>Friedbert Pautzke</td>
</tr>
<tr>
<td>V08/12/04/07</td>
<td>Revise</td>
<td>Steffen Saalmann</td>
</tr>
<tr>
<td>V09/13/04/07</td>
<td>Revise</td>
<td>Joachim Nern</td>
</tr>
<tr>
<td>V10/13/04/07</td>
<td>Revise</td>
<td>Steffen Saalmann</td>
</tr>
<tr>
<td>V11/08/05/07</td>
<td>Rework after PO comments</td>
<td>Tomás Pariente</td>
</tr>
<tr>
<td>V12/09/05/07</td>
<td>Rework after PO comments</td>
<td>Joachim Nern</td>
</tr>
<tr>
<td>V13/10/05/07</td>
<td>Rework after PO comments</td>
<td>Friedbert Pautzke</td>
</tr>
<tr>
<td>V14/16/05/07</td>
<td>Rework after PO comments</td>
<td>Steffen Saalmann</td>
</tr>
<tr>
<td>V15/22/05/07</td>
<td>Final Revise</td>
<td>Steffen Saalmann</td>
</tr>
</tbody>
</table>
## TABLE OF CONTENTS

**EXECUTIVE SUMMARY** .......................................................................................................................... 2  
**DOCUMENT INFORMATION** .................................................................................................................. 3  
**TABLE OF CONTENTS** ........................................................................................................................... 5  

1 **TECHNOLOGICAL BACKGROUND** .................................................................................................... 6  

2 **INFRAWEBS PROPOSITION** .................................................................................................................. 8  
   2.1 INFRAWEBS Framework ..................................................................................................................... 9  
   2.2 INFRAWEBS Conceptual Architecture ............................................................................................. 10  
   2.3 INFRAWEBS middleware and P2P implementation ........................................................................ 13  
   2.4 INFRAWEBS Physical Architecture ............................................................................................... 14  

3 **IMPACT AND BENEFITS FOR POTENTIAL USERS** ...................................................................... 18  
   3.1 Service Providers ................................................................................................................................. 18  
   3.2 Service Composition Providers ........................................................................................................ 19  
   3.3 Service Application Providers .......................................................................................................... 21  
   3.4 Service Application End-users ......................................................................................................... 22  

4 **INFRAWEBS FRAMEWORK MODULES** .......................................................................................... 23  
   4.1 INFRAWEBS Integration Framework .............................................................................................. 23  
   4.2 Semantic Information Router .......................................................................................................... 24  
   4.3 Semantic Web Services Designer .................................................................................................... 25  
   4.4 Semantic Web Services Composer .................................................................................................. 27  
   4.5 Distributed Repository of Semantic Web Services ......................................................................... 28  
   4.6 Organizational Memory .................................................................................................................... 30  
   4.7 Service Access Middleware ............................................................................................................ 31  
   4.8 Semantic Web Services Executor and QoS Broker ........................................................................ 32  
   4.9 Security & Privacy ............................................................................................................................. 33  

5 **PROJECT FACTS** .................................................................................................................................. 35  
   5.1 Availability of results .......................................................................................................................... 35  
   5.2 Contractors Involved ........................................................................................................................ 36  
   5.3 Project Characteristics ...................................................................................................................... 37
1 TECHNOLOGICAL BACKGROUND

The complexity of the technical landscape and the need of new business models that are able to satisfy all the actors in the value chain has brought to us the paradigm of service-oriented computing, whose importance and impact seems to be growing all the time.

According to the Service-Oriented Computing Research Roadmap\(^1\) published in March 2006, organizations face rapidly changing market conditions, new competitive pressures, new regulatory fiat that demand compliance, and new competitive threats. All of these situations and more drive the need for the IT infrastructure of an organization to respond quickly in support of new business models and requirements.

Integration and infrastructure management are the key elements of an on demand operating IT environment. Integration enables the efficient and flexible combination of resources to optimize operations across and beyond the boundaries of an organization and enables them to interoperate seamlessly. Infrastructure management addresses two objectives: automation and virtualization of the environment. Automation of the environment is achieved by the capability to reduce management complexity to enable better use of assets, improve availability and resiliency, and reduce costs based on business policy and objectives. Virtualization of the environment is achieved by the capability to provide easy access to and a single consolidated view of all available resources in a network - no matter where the resources of information reside. Service orientation provides the underlying implementation that makes an on-demand IT operating environment a reality by supporting the functions of both integration and infrastructure management.

Service-oriented computing utilizes services as the constructs to support the development of rapid, low-cost, and easy composition of distributed applications. Services are autonomous, platform-independent computational entities that can be used in a platform independent way. Services can be described, published, discovered, and dynamically assembled for developing massively distributed, interoperable, evolvable systems. Services perform functions that can range from answering simple requests to executing sophisticated business processes requiring peer-to-peer relationships between possibly multiple layers of service consumers and providers. Any piece of code and any application component deployed on a system can be reused and transformed into a network-available service. Services reflect a "service-oriented" approach to programming, based on the idea of composing applications by discovering and invoking available applications to accomplish some task. Services are most often built in a way that is independent of the context in which they are used. This means that the service provider and the consumers are loosely coupled.

Web Services are the current most promising technology based on the concept of Service-oriented computing, even if some problems are still to be solved.

The visionary promise of services technologies is a world of cooperating services where application components are assembled with little effort into a network of services that can be loosely coupled to create business processes and agile applications that span organizations and computing platforms. Services hold the promise of moving beyond the simple exchange of information to the concept of accessing, programming,

\(^1\) SOC Research Roadmap, elaborated by Michael P. Papazoglou, Paolo Traverso, Scharam Dustdar and Frank Leymann
and integrating application services that are encapsulated within old and new applications. An important economic benefit of the Service-Oriented Computing paradigm is that it enables application developers to dynamically grow application portfolios more quickly than ever before, by creating compound application solutions that use internally existing organizational software assets which they appropriately combine with external components possibly residing in remote networks. Previously isolated Enterprise Resource Planning (ERP), Customer Relationship Management (CRM), Supply Chain Management (SCM), Human Resource Management (HRM), financial and other legacy systems can now be converted to service enabled architectures and integrated more effectively than when relying on custom, point-to-point coding or proprietary Enterprise Application Integration technology. The end result is, that it is then easier to create new composite applications that use pieces of application logic and/or data that reside in the existing systems. This represents a fundamental change to the socio-economic fabric of the software developer community that improves the effectiveness and productivity in software development activities and enables enterprises to bring new products and services to the market more rapidly.

Key to this concept is the service-oriented architecture (SOA). SOA is a logical way of designing a software system to provide services to either end-user applications or to other services distributed in a network, via published and discoverable interfaces. A well constructed, standards-based Service Oriented Architecture can empower a business environment with a flexible infrastructure and processing environment. SOA achieves this by provisioning independent, reusable automated business process and systems functions as services, and providing a robust and secure foundation for leveraging these services. Efficiencies in the design, implementation, and operation of SOA-based systems can allow organizations to adapt far more readily to a changing environment.

Services technologies are being shaped by, and increasingly will help to build a, modern society as a whole, especially in vital areas such as dynamic business, health, education, and government services. Applying services technologies leads to reduced complexity and costs, exposing and reusing core business functionality, increased flexibility, resilience to technology shifts, and improving operational efficiency. For all these reasons, it is expected that the Service Oriented Computing paradigm will exhibit a steeper adoption curve, as it solves expensive and intractable business and technology problems, and will infiltrate more of the applications portfolio, than previous application technologies.

Web services define a new paradigm for the Web, in which a network of computer programs becomes the consumer of information. However, Web service technologies only describe the syntactical aspects of a Web service and, therefore, only provide a set of rigid services that cannot adapt to a changing environment without human intervention. Realization of the full potential of the Web services requires further technological advances in the areas of service interoperation, discovery, choreography, and orchestration.

On the other hand, existing software tools and systems are mostly incoherent and oriented on “static” – non-dynamically – web service structures. Mainly this is due to proprietary aspects or features preventing an automated flow of effective information exchange. As a result, existing development and networking platforms are missing important features such as re-configurability and adaptability as well as web service specific aspects like orchestration and choreography.

A possible solution to all these problems is likely to be provided by converting Web services to Semantic Web services (SWS), which are self-contained, self-describing, semantically marked-up software resources that can be published, discovered, composed, and executed across the Web in a task driven semi-automatic way. SWS
can constitute a solution to the integration problem, as they enable dynamic, scalable and reusable cooperation between different systems and organizations.

There are two major initiatives working on developing a world-wide standard for the semantic description of Web services. The first one is OWL-S [OWL-S 2004], a collaborative effort by BBN Technologies, Carnegie Mellon University, Nokia, Stanford University, SRI International, and Yale University. OWL-S is intended to enable automation of Web service discovery, invocation, composition, interoperation, and execution monitoring by providing appropriate semantic descriptions of services. The second one is the Web Service Modelling Ontology (WSMO), a European initiative intending to create ontology for describing various aspects related to Semantic Web Services and to solve the integration problem. WSMO has been under development over the past three years and has been adopted in several Integrated IST Projects such as DIP\(^2\), SEKT\(^3\), Knowledge Web\(^4\), and ASG\(^5\), by consortia including in total more than 50 academic and industrial partners.

At the moment the practical application of SWS technologies is still rather restricted due to several reasons, some of which are:

- the high complexity of both OWL-S and WSMO,
- the lack of standard domain ontologies and unavailability of mature tools supporting WSMO or OWL-S, and
- the absence of pilot applications focusing on every-day needs of consumers, citizens, industry etc., which can demonstrate the benefits of using semantics.

There is then a clear need of tools that on the one hand allow an easy development of SWS, hiding as much as possible the complexity of dealing with semantic descriptions of services, and on the other hand, facilitate the creation of semantic-enabled applications, in order to bring this technology to the mainstream market.

### 2 INFRAWEBS PROPOSITION

The IST research project INFRAWEBS, which has been successfully completed in the beginning of 2007, proposes a technology-oriented step for overcoming some of the above-mentioned problems. It focused on developing a Semantic Service Engineering Framework enabling creation, maintenance, and execution of WSMO-based SWS, and supporting SWS applications within their life-cycle. Being strongly conformant to the current specification of various elements of WSMO (ontologies, goals, semantic services and mediators), the INFRAWEBS Framework hides the complexity of creation of such elements by:

- Identifying different types of actors (users) of Semantic Web Service Technologies;
- Clarifying different phases of the Semantic Service Engineering process, and
- Developing a specialised software toolset oriented to the identified user types and intended for usage in all phases of the SWS Engineering process.

---


\(^3\) [http://sekt.semanticweb.org/](http://sekt.semanticweb.org/)

\(^4\) [http://knowledgeweb.semanticweb.org/](http://knowledgeweb.semanticweb.org/)

\(^5\) [http://asg-platform.org/](http://asg-platform.org/)
In the further course of this section the main features of the INFRAWEBS proposition and framework will be presented in more detail.

### 2.1 INFRAWEBS Framework

The main objective of INFRAWEBS was to develop an ICT framework consisting of several specific software tools, which enables software and service providers to generate and establish open and extensible development platforms for Semantic Web Service based applications.

Conceptually, the INFRAWEBS Framework is a Service-Oriented Architecture (SOA) comprised of coupled and linked INFRAWEBS semantic Web units (SWU), whereby each unit provides tools and components for analyzing, designing, and maintaining WSMO based semantic Web services and SWS applications within the whole life-cycle. A very important aspect, concerning the development and operation of any SOA-based application, is to identify the actors and their roles in the scope of the application. The following actors have been identified as potential users of the INFRAWEBS Framework:

- **Semantic Web Service Provider** – any provider of already existing Web services, who would like to convert it to Semantic Web services and to publish them.
- **Semantic Web Service Broker** (Aggregator) – a provider, who would like to create and publish a service with some desired functionality via composition of several existing Semantic Web services.
- **Semantic Web Service Application Provider** – an organization, that would like to design its own application based on Semantic Web Service Technology.
- **Web Service Application Consumer** – an end-user of a Web Service Application, who transparently uses the INFRAWEBS Framework (while using the Application) for finding and executing a Web service or a composition of Web services able to satisfy his/her request (goal).

The proposed categorization of INFRAWEBS Framework users allows to identify more clearly the set of different tasks that the Framework is able to accomplish, in order to satisfy the objectives of these users, as well as the set of necessary components. It should be noted that this categorization is tentative and in some cases an organization or a person can perform tasks that are specific for more than one role. For example, an organization that develops a semantically-enabled application may perform tasks that are associated not only with Semantic Web Service Application Providers but also with SWS Providers, e.g. semantically describe Web services provided by the organization itself, and with SWS Brokers (Aggregators), e.g. compose several SWS to provide a value added service for its customers.

The INFRAWEBS Framework is developed to support all stages of the semantic Web service life-cycle presented on Figure 2-1.
This life-cycle encompasses the following phases:

- **SWS Creation** – combines activities related to creation of semantic descriptions of the Web services, as well as of necessary ontologies and goals. Created descriptions are then persistently stored and some of them published for common usage.

- **SWS Composition** – already created semantically described services can be combined (by their provider, or an external integrator) to provide new, value-added services. Composed services are also stored as semantic descriptions and can be further used just like other services.

- **SWS Discovery** – enables already described services to be discovered for usage. Discovery is a semantically-enabled process of matching user requests (specified as WSMO goals) to service functionality (represented in the WSMO service capability).

- **SWS Selection** – allows users (be they human or other services) to choose in a pro-active manner which of the discovered services to be executed. Generally, selection can also be done automatically, for example by the discovery agent (tool), but INFRAWEBS supports also scenarios requiring user selection or domain-specific automated selection provided by semantically-enabled applications, and for this reason a dedicated step in the SWS life-cycle is necessary.

- **SWS Execution** – deals with the actual Web service delivery, when the user provides some input to the service and gets it running to obtain expected results.

- **SWS Monitoring** – an important phase when information about executed services is gathered to be further used for service selection.

### 2.2 INFRAWEBS Conceptual Architecture

In order to satisfy the requirements of the identified types of the users the INFRAWEBS conceptual architecture consists of two main elements: INFRAWEBS SWU Design-time Shell and INFRAWEBS SWU Run-time Environment (see Figure 2). All SWU components are organized in two directions: 1. Problem solving based on semantic information (or Logic-based problem solving) versus problem solving based on non-semantic information (similarity-based problem solving) and 2. Different types of information needed for both kinds of problem solving.
From these points of view the **Semantic Web Unit** proposes a set of tools aimed at different purposes:

1. **Information structures** for storing and retrieving semantic and non-semantic data:

   - **Distributed Semantic Web Service Repository/Registry (DSWS-R)** enables effective storage and retrieval of all elements of the Semantic Web (WSMO objects): goals, ontologies, semantic Web services, and mediators, written in the WSML language. Each DSWS-R consists of two parts: Local Repository – a place where all WSMO objects created in this Unit are stored; and Local Registry – a place where publicly available (i.e. published) advertisements of WSMO objects are stored. Advertisements contain the minimal necessary information needed to locate the advertised document: an identifier of the described WSMO object, the endpoint of the owner Registry, as well as the policy for propagating this advertisement. Advertisements are published in the local Registry by their owner and are then propagated to partner Registries depending on their policy. Partnership characterizes the willingness of service providers to distribute advertisements of its own WSMO objects to Registries of its business partners. It is expected that partnerships will be actually defined by business contracts between organizations in practice. The DSWS-R also stores some additional non-semantic data, e.g. graphical models of logical expressions defined in all WSMO descriptions stored in the Repository.

   - **Semantic Information Router (SIR)** contains annotation meta-data about Web services (WSDL files) used for grounding of SWS created in a local SWU and for their subsequent execution. SIR provides GUI for Web services annotation and categorization.

   - **Similarity-based OM** contains a special representation (so called “knowledge objects”) of both semantic and non-semantic objects stored in other INFRAWEBS information structures (DSWS-R and SIR). OM is a Web service...
implementation of the INFRAWEBS case-based memory, allowing for effective retrieval of objects based on their content.

2. **Tools for creation and maintenance of both semantic and non-semantic data:**

- **SWS Designer** enables design of WSMO-based semantic Web services on the basis of existing non-semantic Web service descriptions stored in the SIR and WSML ontologies, as well as by reusing already existing WSMO semantic service descriptions stored in the DSWS-R.
- **SWS Composer** is a tool for design-time creation of new Semantic Web services through composition of other existing WSMO-based semantic Web services stored in the DSWS-R.
- **Goal Editor** provides means for creation of WSMO-based goal templates used for the design of SWS-based applications and also used at run-time for creation of specific application user goals.
- **CBR Recommender tool** is a similarity-based tool facilitating operation of all INFRAWEBS “semantic-based” components by utilizing “past experience”. Knowledge objects stored in OM are used as case descriptions analyzed to determine the most similar solution to the current problem.

3. **Problem-Solving Methods used for creating and maintaining Semantic Web Services:**

- Logic-based discovery
- Application-specific decision-support methods used for service composition, compensation, monitoring etc.
- Several methods for calculating similarity – structural, linguistical, statistical, etc.

**SWU Run-time Environment** is responsible for communication with different INFRAWEBS users and other semantic Web units, ensuring security and privacy of these operations. The Run-time Environment consists of the following tools:

- **Service Access Middleware (SAM)** provides a retrieval and execution interface for advertised SWS. The user mandates a user interface agent for fulfilling the service demand and the agent provides recommendations based on the user's query. The matchmaking between user request and service capability are similarity- and logic-based.
- **SWS Executor** module processes SWS descriptions (in WSMO) using choreography and orchestration engines for executing specific SWS related rules.
- **QoS (Quality-of-Service) Monitor** provides functionalities for monitoring the SWS execution process by feeding back extracted metric data.
- **Security and Privacy** enabler realised as an artificial “immune defence system” allowing the INFRAWEBS framework to function safely under changing conditions.

The INFRAWEBS Conceptual Architecture reflects a novel approach for solving problems occurring during in the process of creating and maintaining semantic Web services and SWS applications. It is based on tight integration of similarity-based and logic-based reasoning. Similarity-based reasoning is used for fast finding of approximate solutions, which are further concretized by the logic-based reasoning.
2.3 INFRAWEBS middleware and P2P implementation

The INFRAWEBS Framework has been implemented on the top of an extensible Enterprise Service Bus (ESB) middleware that exposes the public methods of the INFRAWEBS components and can be extended by any future components or services. Such Integrated INFRAWEBS Framework (IIF) can be seen as the underlying infrastructure for communication and integration of all the INFRAWEBS components, and at the same time the unique selling point for exposing the functionality of such components to the external world in form of services.

The INFRAWEBS platform consists of a decentralized network of nodes or peers where every peer has a similar role and no one peer is singular. All peers are connected with the rest of peers through the IIF infrastructure.

Each IIF peer consists of a centralized bundle of components within a single server where there is a central IIF Server component and several INFRAWEB components. These INFRAWEBS components have been developed in different technologies, as Java or .NET. The IIF do not offer the same facilities for the integration of components based on .NET and Java technologies, basically because the IIF is a java-based framework, but it offers Web Services'-based technology, which allows the easy incorporation of any component that allows the use of this standard technology.

The IIF is delivered as a peer-to-peer distribution. Figure 3 shows this component distribution for a single peer, a logical view of this peer showing the possibilities of integration of components with different technologies within the peer:

![Figure 3 IIF one-peer architecture](image)
The IIF can be set up as a peer-to-peer distribution. Every peer can be deployed containing all or part of the set of INFRAWEBS stack of components. The IIF hides to the different INFRAWEBS users the complexity of dealing with this p2p architecture.

Figure 2-3. IIF peer-to-peer architecture

2.4 INFRAWEBS Physical Architecture

The next figure shows a general view of the physical architecture and communication between the different components of the IIF, highlighting in different colours the work package in which each component has been developed.
During the design-time phase the required semantic descriptions are created. These descriptions include ontologies, goals, and semantic web services. Such semantic descriptions enable computers to reason about the data and the processes themselves. **Design-time** components are the SWS-D, SWS-C, DSWS-R, OM, SIR and SAM, as shown in the figure below:
The run-time phase involves interaction with the client and execution of the semantic web services. Henceforth, not all components involved during the design time phase are involved during runtime. Run-time components are mainly SAM, SWS-E, QoS Monitor and Security, although other components like the DSWS-R or the OM are used for example in the context of the service discovery.
Figure 6: Runtime Phase of the INFRAWEBS Framework
3 IMPACT AND BENEFITS FOR POTENTIAL USERS

In the quest of competitive edge, companies in Europe are pressed to gain innovation, become faster and more flexible (i.e. highly dynamic and adaptable), but also to offer a wider range of stable and reliable services along with a personalized interaction with customers, clients, and partners.

Undoubtedly, ICT supporting modern, dynamic, reconfigurable, and scalable technologies like the INFRAWEBS approach do play an important role in tackling these challenges and implementing advanced semantically based knowledge domains.

INFRAWEBS is a flexible, interoperable, and reconfigurable framework, enabling organizations to build up partnerships faster and in a more effectively way with respect to the service generation, execution, and distribution process. By allowing peers to change their role – to be client, broker, and service provider within one environment – INFRAWEBS ensures a highly dynamic and efficient service production process and workflow, which spans the whole service lifecycle.

The impact for INFRAWEBS potential users is pointed out in the following subsections.

3.1 Service Providers

In INFRAWEBS a semantic Web service can be created in two ways– by converting an existing non-semantic Web service into a semantic one (this process is called “SWS design”) or by composition of several existing semantic Web services. This section describes the process of creation of semantic Web services from existing non-semantic ones. This process is realized by creating semantic description, conforming to the WSMO specification, on the basis of existing WSDL descriptions as well as by reusing already created semantic descriptions of similar services.

The life-cycle of the SWS design process is defined by both the static structure of the semantic description of a SWS, which is fully determined by the selected standard for SWS representation – WSMO, and certain assumptions on what kind of additional semantic information is needed, where it is stored and how it can be found. From this point of view, we assume that non-semantic Web services are implemented and formally described (as WSDL definitions) outside the INFRAWEBS Framework and are stored in some UDDI-like repositories.

In order to facilitate finding of such Web services, the service provider can annotate them with natural language based metadata (for example, according to the Dublin Core6 schema). It is necessary to emphasize that the addition of such metadata does not convert a Web service into a WSMO-based semantic service. It simply enables the process of finding the annotated service by users or business partners. In the INFRAWEBS Framework this service annotation activity is considered as the first, preliminary step of the semantic service design process and it is seen as a process of registration of Web services into the Framework.

6 http://dublincore.org/
The INFRAWEBS Framework supports this activity through a dedicated component – the Semantic Information Router (SIR). The Service Provider is able to annotate the Web services using the SIR component, as it is explained in section 4.2. The SIR is a platform for a meta-data based content management and aggregation, that was used in INFRAWEBS for registration and annotation of Web Services (and more precisely their WSDL definitions), which are then used in the process of SWS design.

The next step of the SWS design process is the creation of its WSML description according to the WSMO framework. However, in order to facilitate this very complicated activity we have split it into several sub-steps:

- Finding a desired non-semantic Web service (described by an annotated WSDL file).
- Finding a set of appropriate WSML ontologies to be used for semantic reformulation of the main Web service functionality in ontological terms. In the INFRAWEBS Framework ontologies are stored in the WSMO object repositories (DSWS-R).
- Creating the semantic description of the service behaviour. According to WSMO, that description consists of service grounding, determining the correspondence between data structures in the semantic and non-semantic descriptions of a WSMO service, and the service choreography describing how it is possible to communicate with the semantic service in order to execute properly the non-semantic Web service grounded to it.
- Creating the semantic description of service functionality, advertising what the service can do. Such semantic service capability description is used for discovery.
- Publishing service descriptions. In the INFRAWEBS Framework the description of semantic services (as well as other WSMO-based semantic object such as ontologies, goals, etc.) are stored in the local (belonging to concrete local SWU) repositories (DSWS-R). The DSWS-R component is then responsible for propagating the advertisement of published objects within the INFRAWEBS Framework.

All these sub-steps (except the last one) can be done in the INFRAWEBS by means of a special tool – the SWS Designer, which is described in more detail in Section 4.3. The publishing activity is performed by the Designer and DSWS-R components.

### 3.2 Service Composition Providers

INFRAWEBS enables Web service aggregators to compose, in design-time, already existing SWS descriptions in order to provide new value-added services. Composed services are also defined as WSMO services and can then be discovered and used in the same way as atomic services. Once there are several SWS, the problem of service composition arises in order to achieve complex user goals. INFRAWEBS provides for users, attempting to create more complex services, a tool to generate these services in a graphical way.

The INFRAWEBS Design-time Composer (SWS-C) deals with the combination of different semantic services to obtain a new complex Semantic Service. It is based on the ontological confirmation of contexts. The workflow methodology is adapted for creating WSMO-based service compositions for the determination and visualisation of data and control flow within the composed service.
The SWS-C composer works in a graphical environment to compose Semantic Web Service accessing already existing semantic components.

The graphical environment consists of:

- a visual tool to edit and modify the service diagram,
- visual tools for importing/exporting services and ontologies,
- visual tools for editing interconnections between services.

Each service is represented as a box marked with the name of the service. The connections between services are given by two types of lines: data flow and control flow. This separation is imposed by the need to construct and edit different aspects of data described in capabilities and interfaces of Web services participating in the composition.

Services that are potentially relevant for the requirements of the service application provider are accessed either through the INFRAWEBS logical discovery by ontological conformance or through the INFRAWEBS similarity-based retrieval. The result of the composition is stored as a WSML description of the WSMO service and associated to the WSMO goal in the DSWS-R. The WSMO-based description of the composed service contains the following information:

- Title – specified by the composition designer (Web service broker or application service provider).
- Non-functional properties – can be specified through the designated editors.
- Imported ontologies – a set comprising of the ontologies used by each service in the composition. In addition, other ontologies can be included in the composition description if necessary.
- Capabilities - the composed service combines the functionality of the included services, hence, its capability is defined as a compilation of the logical expressions defined the preconditions, postconditions, assumptions and effects of each service. As the composed service may have additional constraints defining the interaction of services or have less preconditions or assumptions (as some inputs are provided by included services), the SWS-C enables editing of the composed capability.
- Interface description – specifying choreography and orchestration parts. The choreography is again defined as a compilation of the separate service choreographies. The sequencing of services is modelled by reordering of the choreography rules (actually by editing and reordering the control states they use) defined in the composed choreography.

At present, WSMO orchestration is still under development, it is yet in the stage of language design and specification. For this reason, the design-time compositions are described as a choreography combining the choreography descriptions of participating services. In the future, when orchestration specification is fully developed, the SWS-C can be extended to create orchestration descriptions on the basis of the workflow description.

“Control flow” and “data flow” definitions are important elements of the composition. "Control flow" definitions influence the interface description of the newly created Semantic Web Service description, as described above. The "data flow", on the other hand, specifies attribute mappings between concepts used as inputs and outputs of the participating services. Attribute mappings are then reflected in the choreography description as new rules that update a state so that data constraints are enforced.

An example for data constraints is taken from the eGovernment use case. The composed service combines a waste tax payment certifying, a sanitary clearance and a
service providing notifications for registration of the new shop. In this case, the registration notification service requires as input, among other parameters, the certificate numbers produced by the other two services, or given by the user. The choreography definition of this service checks only whether a valid request is passed to the service, but does not explicitly check the necessary certificate numbers. In order to capture the dependency, data flow mechanisms are used in the composition. They are then reflected as a new rule which explicitly updated the new business notification service request with the certificate numbers provided as result of the other two services.

3.3 Service Application Providers

One of the main objectives of INFRAWEBS is to support organizations that want to create semantically-enriched applications based on Semantic Web Services. In the previous sections the support for the creation of SWS descriptions via service design or composition has been explained. They will also be used by Semantic Service Application Providers to create the basic functionality of their applications. However, in order to use these services the application should be able to explore all possibilities provided by the run-time INFRAWEBS Environment. At run-time, discovery of a semantic Web service is done via matching a given user goal description against all service capability descriptions, hence, the first task of the application provider is to prepare a mechanism for creating the goals.

The development of semantic-enabled applications is an almost new field, therefore regular developers do not possess specialised expertise. In order to create these applications, the service application providers have to perform the following activities:

- Application design activities: goal template creation, configuration of similarity metrics.
- Application implementation.

INFRAWEBS offers a set of components to fulfill this twofold approach:

- A Goal editor, included in the SWS Designer, which allows defining goal templates in a graphical way. The graphical definition of WSMO goals is a further positive aspect of INFRAWEBS due to hiding the complexity of WSMO.
- The INFRAWEBS semantic ESB framework of integrated components enables to develop this type of applications that allows the developers:
  - the formulation of user goals,
  - the discovery of matching services,
  - the selection of services to be used, and
  - the execution of selected services.

By using the INFRAWEBS IIF it is possible to create applications handling Semantic Web Services without “touching a line” of WSML code.

The INFRAWEBS components used by the service application providers are:

- the SWS Designer (goal editor),
3.4 Service Application End-users

End-users benefit of the semantic-enabled applications delivered by the application providers using the INFRAWEBS Framework.

As discussed above, semantic service application functionality is provided by the Semantic Web Services included in its architecture. In the process of running such applications the INFRAWEBS Run-time Environment is used to provide support for the following activities:

- Refinement of goal templates: during the design of the application, goal templates have been defined to represent the most common user requests to the application. These goal templates are included in the application UI as specific forms where necessary data has to be entered by the user in order to use a certain functionality. When the user fills in these specific fields, e.g. the desired destination for a flight, the application uses INFRAWEBS to produce a concrete goal instance from the particular goal template.

- Service discovery: On the basis of the concrete goal, discovery compiles a ranked list of services that match this goal.

- Service selection: The application enables the user to select a suitable service from the list of matching services.

- Service execution: Finally, of the selected Semantic Web Service can be executed to provide its functionality and the results are appropriately presented to the user.

In INFRAWEBS these activities are performed through the Service Access Middleware (SAM), which provides a single API for the semantically-enabled applications to use. The methods of the SAM are accessible via the IIF implementation.

The viability of the INFRAWEBS approach has been tested by two pilot applications with high level of complexity. The first application is the STREAM Flows! System (SFS), in which customers can create and reuse travel packages. The application is built upon a Service Oriented Architecture and uses the INFRAWEBS framework to discover access, compose and invoke Semantic Web Services involved in Travel Packages. The second application is based on an eGovernment scenario. It illustrates interactions carried out by semantic Web services in the scope of public administration and among these, citizens and enterprises with emphasis on the E2A (Enterprise-to-Administration) integration. These demonstrators have shown some of the possible enhancements semantic-enabled applications can provide, such as dynamic composition of travel packages or static composition on demand of eGovernment services.
The INFRAWEBS Integration Framework (IIF) middleware, introduced in section 2.3, is a network of interconnected components using WS technology. It constitutes a Service Oriented Architecture (SOA).

The INFRAWEBS ESB manages automatically the execution and shutdown of components and is able to check if they are alive or not. The framework is based on Java and gives support for integration of Java-based components and non-Java based components. It consists of an IF Connector component that is instantiated by Java-based components to permit them to exchange messages to each other and an IF Server that routes those messages to the appropriate target manages the configuration of the whole INFRAWEBS Development Environment and its proper execution. An IIF Server exposes a complete Web service implementation that collects the full set of operations supported by the components plugged within the IIF. This Web service can be used by non-Java components to send messages to other components plugged within the INFRAWEBS ESB infrastructure.

INFRAWEBS provides two ways of communication for Java-based components:

- The specific way: by instantiating a specific INFRAWEBS ESB “delegate” for a particular component's interface to which a message has to be sent. The delegate 'knows' to which component the message has to be sent and also the details about "how to build" and send the message.

- The generic way: INFRAWEBS provides methods to generically invoke a remote method (sending a message to a remote component), it means, the user just has to know the name of the method in the component and the parameters required by this method (a list of available messages are published in the annex of this document).

For both ways of communication the INFRAWEBS ESB supports different kinds of messages (Message exchange patterns, MEP):

- One way: just send a message without expecting any result from the component.
- In/out Synchronous: send a message to a component and wait until the result is received from the component. This message sent is blocking
- In/out Asynchronous: send a message to a component and the result is retrieved later via a call-back. This message sent is not blocking
- Publication-subscription: a component subscribes itself to a message that belongs to a set of messages that can be published.

The INFRAWEBS IF Connector is a Java client-side API that can be used for sending and receiving all types of messages in both specific and general ways of communication. For non-java components or applications, the Web service interface of the IIF shall be used instead.
The IIF is able to manage automatically the execution and shutdown of its components as well as to check if the components are alive or not. Since the IIF consists of a set of different components and tools for creating, maintaining and executing WSMO-based Semantic Web Services, it can be considered as a Semantic SOA and the use of an underlying ESB middleware provides the IIF with the necessary extensibility to be a Semantic ESB.

### 4.2 Semantic Information Router

Semantic Information Router (SIR) is a semantic content and service management product, which is able to manage rich content and Web Services descriptions in a single solution. SIR lets service developers register their Web Services descriptions into a centralized repository with digital content that makes it re-use more efficient. SIR can be integrated to existing portal applications and business processes with open standards.

The SIR is a platform for a meta-data based content management and aggregation, is used in INFRAWEBS for registration and annotation of Web Services (and more precisely their WSDL definitions), which are then used in the process of SWS design.

The SIR registration interface is a Web interface for human beings who wish to register, annotate and categorize WSDL and BPEL files to the INFRAWEBS system. This interface is typically used by a Web service provider to enable finding of this Web service definition, within or outside of the scope of the provider organization.

The SIR registration interface can be considered as the entry point to the overall INFRAWEBS system, since the phase of SWS Creation typically starts by finding the appropriate WSDL file. This interface has been developed to support the roles of:

- Developers who can register, annotate and register WSDL and BPEL files.
- Administrators who can manage registered services and metadata schemas used to annotate the services by the developers.
- Semantic service engineers (at the SWS provider organization) who query the SIR in order to find appropriate WSDL definitions of the services they want to described semantically.

Summarizing, SIR is a software product integrated within the INFRAWEBS platform providing functionalities for:

- registering and annotating WSDL files to the INFRAWEBS system,
- notifying the INFRAWEBS system about new Web Services,
- categorization of WSDL files under service categories for easier retrieval by other design-time components such as the Designer of the INFRAWEBS system,
- retrieval of WSDL files and their metadata properties via the SPARQL query language.

SIR stores metadata, about the registered WSDL and BPEL files, natively in the Resource Description Framework (RDF) format. This metadata is exposed for querying via Simple Protocol And RDF Query Language (SPARQL).
4.3 Semantic Web Services Designer

The conceptual architecture of the INFRAWEBS Designer is shown in the next figure:

![Conceptual Architecture of the INFRAWEBS Designer](image)

**Figure 7: SWS-D Conceptual Architecture of the INFRAWEBS Designer**

*Temporary Store* is an abstract module representing a place, where all local files produced or used by the INFRAWEBS Designer are stored. Such files include a current WSML description of a semantic service under design, WSML descriptions of all ontologies used for the service description, files with graphical models of WSML logical expressions used for representing the service capability and choreography etc.

*Navigator* is an abstract module centralizing an access to other modules of the INFRAWEBS Designer. It stores to and retrieves from the Temporary Store all locally stored files and switches the control between the Communicator and different INFRAWEBS specialized editors. The Navigator is also responsible for creating a so called “Service Tree”, which may be seen as an initial skeleton of a newly created semantic service description. The whole service designing process in the INFRAWEBS Designer is organized as “populating” this tree.

*Communicator* is responsible for communication with the user and the external components as the OM (Organizational Memory), the DSWS-R (Distributed Semantic Web Service Repository) and the SIR (Semantic Information Router).

*Publisher* is a module responsible for storing a semantic service description in an external (for the Designer) storage – the INFRAWEBS DSWS-R. In order to guarantee the correctness and completeness of such a description the Publisher validates...
different parts of the description and finalizes it with some information needed to insure the proper use of this service in the future.

**WSMO Service Editor** is a module responsible for creating and editing the WSML description of a semantic Web service according to the WSMO Framework. The Editor combines three specialized graphical ontology-driven editors for creating and editing different parts of a description as well as a specialized module for in-memory loading, visualization, and manipulation of WSML ontologies used by these editors:

- **NFP Editor** is responsible for graphical creating and editing the WSMO-based description of non-functional properties of a semantic Web service.
- **Capability Editor** is responsible for graphical creating and editing the capability description of a semantic Web service. The Editor is an extension of the INFRAWEBS Axiom Editor, which allows graphical creating, editing and reusing of complex WSML-based logical expressions.
- **Choreography Editor** is responsible for graphical creating and editing the description of the choreography of a semantic Web service, which consists of an ontology-based state signature and state transition rules.
- **Ontology Store** is responsible for loading, visualizing and using WSML ontologies. It provides the basic graphical elements corresponding to the WSMO ontology concepts, instances, and relations for creating graphical models of a semantic service description, supports automatic on-demand loading of the required ontologies as well as makes the actualisation of the ontology concept structure during the process of the graphical model creation.

The necessity of intensive development of tools supporting semantic Web service technology, and especially ones based on the WSMO framework, has been clearly recognized in WSMO community. However, the analysis of existing tools for creating WSMO object has shown that they are mainly oriented to researchers and developers, working in the area of Semantic Web Services technology, rather than to the real “regular” users of such a technology – Web service providers, semantic service brokers and semantic application providers. That very end-user orientation of the INFRAWEBS Designer is the basic feature making the product different from all similar research products developed in the area of Semantic Web Services.

The main outcomes of the Designer are the following:

- As a tool for automatic generation of programs (descriptions of WSMO services and goals written in WSML language), the Designer always produces syntactically correct programs that passes all checks applied by WSMO Validator and WSML Parser. Moreover, the auto-generated WSML programs are semantically consistent both with regard to WSML ontologies used for their creation and with regard to the last WSMO specification for WSMO objects described by these programs. The evidence for this conclusion is the ability to use the produced WSML files both for service discovery and for execution.

- As a tool oriented towards end-users of semantic technology, who are not familiarized with the use of the WSML language, it can be concluded that the proposed approach for graphical creating the WSMO objects is understandable and usable. Guided by the Designer the users are able to create rather complex (from logical point of view) axioms and transition rules without any knowledge of WSML syntax. The developed GUI is sufficiently intuitive and easy learnable. The available set of primitives (the Designer operations) is adequate for constructing and editing descriptions of goals and services used in the test beds.

- The developed model of the process for graphical creation of WSML expressions is flexible enough and allows the use of alternative ways for constructing the same
expression, depending on the experience and the way of thinking of the concrete user of the Designer.

- The developed mechanism for reusing the existing descriptions of WSMO objects via “copy-and-paste” of graphical models is understandable to the users and significantly reduces time and efforts for creating new objects.

- The integration of the Designer with facilities for creating WSMO goals has made it a powerful instrument for creating composite goals and significantly simplifies the work of the service application Designer.

- The Designer can be effectively used for validation of the semantic correctness and completeness of WSML descriptions of WSMO services and goals created without using the Designer. The ontology-driven approach used for creating logical expressing in the Designer, allows identification of all erroneous usages of names of ontologies, ontology concepts and concept attributes, as well as errors caused by improper use of the concept types. Moreover, the Designer is able to find errors and incompleteness in the description of imported ontologies used by the object as well as in description of the service interface. The problems discovered in a “third-party” WSML file can be easily identified based on error messages issued by the Designer and the graphical representation of logical expression, which is automatically produced by the tool.

- The Designer can be effectively used even for validation of the semantic correctness and completeness of WSML descriptions of WSMO ontologies. For example, the Designer built-in mechanism for on-demand loading of ontologies enables identifying the incorrect use of ontology names, imported by the concrete ontology, the lack of the cited ontology in the workspace as well as an incorrect use of concept attributes inconsistent with attribute inheritance rules. All these features of the Designer make it a powerful tool not only for novices in WSML but for WSML experts as well.

### 4.4 Semantic Web Services Composer

The INFRAWEBS Design-time Composer (SWS-C) deals with the combination of different already created semantic services to obtain a new service. The SWS Composer provides an ontology-oriented service-based workflow of WSMO-described services.

The SWS-C is used in INFRAWEBS by the application service providers.

The similarity-based Organisational Memory (OM) supports the search for compatible services by filtering and suggesting services that use lexical and ontological similar context. The Organisational Memory provides support for making decision on the base of similarity inference in the case the composer tries to combine services from different application domains. Services from different domains usually use different sets of ontologies. But composition of these services is still possible, because they use some general domain independent set of ontologies and ensure semantic consistency of the composition. The services, that can be combined, are supposed to use similar semantic annotations connected to the domain ontologies or meta-data, etc. and their annotations match with some measure of similarity.

Logical discovery is necessary for searching of ontologically interconnected services; these are services with common ontologies set. Discovery of matching services is realized by querying the Service Agent Middleware (SAM) for searching a semantically
suitable service. The query is constructed as a WSMO goal from the description of selected service in the composition.

The data integration within the SWS-C is realized by

- Referencing shared variables.
- Importing service ontologies.
- Matching ontological attributes from postconditions and precondition of participated services.
- Constructing additional logical relations and logical constraints for the composed semantic service.

Visualization of the service composition is organized by

- Graphical tools for importing/exporting services and ontologies.
- Graphical tools for editing and modifying data and control flows.

The SWS Composer works in the development environment with a graphical user interface to compose semantic web service components in accordance with WSMO specification. It is naturally that the composed service may have additional constraints defining the interaction of services. The Composer gives means for defining such constraints that are logically expressed in the precondition of the whole composed service. The result of the composition is a WSMO object that can be stored in the DSWS-R, retrieved by the SAM and executed by the SWS-E.

Adding a service to the composition is supported by

- using INFRAWEBS logical discovery,
- INFRAWEBS OM-based Similarity search, and
- browsing and retrieving services from the local repository.

The services, which best matches the composition needs are selected by the user and connected using a workflow-like WSMO Studio plugin.

### 4.5 Distributed Repository of Semantic Web Services

The role of the DSWS-R component is to enable persistent storage and retrieval of semantic descriptions as well as their publication and propagation within the p2p network of INFRAWEBS units. The DSWS-R component provides both a local repository, where service providers and application providers can store their service, ontology and goal descriptions, and a registry component which enables public usage and access to the descriptions that their owner wants to share within the INFRAWEBS framework.

The DSWS-R functionality is used in design time to store ready descriptions and to provide access to previously stored WSMO descriptions that will be used in the process of designing or composing new services. Hence, the DSWS-R is basically used to provide efficient access, storage and lookup of WSMO descriptions.

In run-time, the DSWS-R is used to retrieve locally or remotely stored descriptions which are to be used in the process of discovery or execution of services. Application providers also use the DSWS-R in run-time in order to retrieve specific goal templates (atomic or composite) that correspond to the application user request. These templates
are then filled in with data provided by the user and are used for discovery of matching services.

No GUI is supported for the run-time access to the DSWS-R, instead the Java interface of the component is used and access is implemented through the IIF.

The DSWS-R is working with semantic descriptions not only of services, but also of user goals and ontologies. Thus, in addition to services, ontology and goal descriptions can also be exchanged and reused by different INFRAWEBS framework users.

In design-time the DSWS-R functionality is visually used through a GUI, provided by the DSWS-R plug-in for WSMO Studio. This plug-in is further integrated with the SWS-Designer so that the user (service creator) can store, update and load semantic descriptions with a click of a button.

The DSWS-R plug-in also provides an easy view over the content of the repository, intuitive browsing functionality and one-click publishing features, as well as registry content viewing. This plug-in supports working with multiple repositories (and registries), which can be useful for providers of big or diverse applications, where it might be suitable that services are grouped according to their purpose.

In INFRAWEBS the WSMO Studio is used to provide the basic functionality and infrastructure for working with WSMO descriptions, and both the SWS-D and SWS-C components are integrated with it. This was one of the reasons to extend the WSMO Studio Repository plug-in and provide users with a GUI for working with the DSWS-R component.

In order to enable the visual storage, retrieval and lookup of WSMO descriptions persisted in the Repository by the SWS-C and a SWS-D user, the DSWS-R plug-in provides functionality for:

• Intuitive storage by drag-and-drop of descriptions residing in the Editor workspace to the Repository (also accessible as menu command);
• Browsing of the Repository content and easy viewing of stored descriptions;
• Removal of stored descriptions;
• Wizard guided export of stored descriptions into the SWS-D or WSMO Studio workspace;
• Visual access to the Registry and one-click publishing process.

We can summarize that Semantic Application Providers and Service Providers use the DSWS-R functionality to:

• Store the ontologies that are necessary for the creation of WSMO descriptions;
• Store the predefined user goals that the application supports;
• Store the semantically described Web services they provide;
• Retrieve any locally or remotely stored semantic description that is necessary for the service or application functioning. Note that only published remote descriptions are accessible;
• Publish any semantic description that can be shared with other users of the INFRAWEBS framework, as well as any semantic Web service that is exposed for users to discover and invoke.

In addition, the DSWS-R will transparently (to these users) do:

• The propagation of advertisements of published descriptions within the INFRAWEBS framework;
• The persistence of graphical models associated with stored descriptions.

### 4.6 Organizational Memory

The OM module represents a similarity based organizational memory and case based recommender tool. The component classifies the system knowledge objects using fuzzy logic techniques as well as statistical and linguistic algorithms as fuzzy matching functions.

The OM module is devoted to the organization of all non-semantic as well as specific semantic (knowledge) objects (WSMO) handled within the SWS development and life cycle. It supports the creators of Semantic Web Services (Semantic Web Service Provider, Service Application Provider) to retrieve relevant or adequate information for the semantic modelling of semantic web services oriented on the WSMO specification. This is achieved by indexing all available textual documents and providing these indexes to the relevant modules.

As already mentioned WSMO objects represent semantic web services, goals, ontologies and mediators written in WSML language. The WSMO community ([http://www.wsmo.org](http://www.wsmo.org)) gives a full overview of the structure of WSMO objects.

Within the integrated INFRAWEBS framework the OM module serves as a case based memory to the following INFRAWEBS modules:

- SWS Designer,
- SWS Composer, and
- Goal Editor.

The OM module is also used by the Discovery module in the first step discovery in the Services Access Middleware (SAM9).

For this purpose the OM module offers functionalities for retrieving web WSMO objects by different methods:

- by structured ontology based queries;
- by unstructured natural language queries;
- by examples.

Searching for WSMO objects by structured query is achieved by endowing the OM module with capabilities for structured query based on WSMO ontologies terms. This query is composed considering different criteria like Web Service preconditions, Web service postconditions etc.

This special retrieval method for WSMO objects is appropriate for the process of designing "new" WSMO objects based on similar existing one. The composition of the query is done by selecting ontology terms out of stored term fragments in the system ontologies.

Natural language queries are a powerful, rapid and easy to use method for retrieving WSMO objects. In this case the user is supposed to provide a description in natural language regarding the demanded WSMO objects. The OM module matches this natural language description to the classified WSMO objects. For this purpose the OM module uses complex lexical methods based on lexical similarity between the words (terms) and their contextual meaning.
The queries by example allow the user to retrieve WSMO objects simply by providing existing WSMO description or fragments of WSMO descriptions. This kind of query is very simple to compose for the user. The OM module parses the example input and retrieves all similar WSMO objects stored in the INFRAWEBS base.

The OM module represents a specialized Organizational Memory for semantic (WSMO) objects. For this purpose fuzzy methods for clustering and classification are applied to WSMO based objects descriptions.

The application of non-semantic (similarity based) reasoning to the semantic objects is a powerful approach for user support. This also facilitates the logic based reasoning of some of the INFRAWEBS components.

For realization purposes of the described queries the OM module implements a novel approach for multilayered classification of objects. Each classification layer has specific dictionaries, filtering methods and similarity calculation methods. WSMO objects are classified in every layer by different criteria.

For complex queries the OM module supports relations between different layers.

### 4.7 Service Access Middleware

The Service Access Middleware (SAM) is the centralized front-end for application developers for discovering, selecting and executing Semantic Web Services. All the functionality needed to use a SWS is available through the SAM. SWS discovery and selection are implemented inside the module, supported by 2 external modules, the DSWS-R (where semantic descriptions are stored) and the OM (used as an indexer). SWS execution is implemented in the SWS-E module. SAM only mediates and supports the communication with SWS-E.

SAM provides a one-stop-shop access point as an abstraction for using SWS. SAM can be used through a SOAP interface for the following goals:

1. **Construction of a goal**: The goal describes what the user wants to achieve by SWS execution. Desired effects are described as logical expressions. Facts and preferences of the user can also be given as additional knowledge.
2. **SWS discovery**: a list of appropriate web services is provided for the goal. It is not guaranteed that any of the listed services can fulfill the desired goal. Some additional information about the listed services may be presented to the user in order to help her in the next step. This information may be related to the technical performance and quality of the web service or to the quality and satisfaction with the real service (usually collected as user feedback).
3. **SWS selection**: the user or the application has to select a service if there are more services available for the given goal. This is also called matchmaking, as it may include preliminary inquiry and investigation of offers and service conditions. Therefore, this step is somewhat merged with the next step of service invocation.
4. **SWS execution**: the selected web service is started for the given goal. The model of the possible sequences of interactions with the user is called choreography. The invoked web service executes its communication pattern with the client (choreography), and it may also communicate with other web services (orchestration) to produce the result.
Goals and results are expressed in WSMO, which can be processed using the wsmo4j toolkit written in Java.

Technological innovation is the use of WSMO in discovery and execution. Business innovation is the semantic access to web services, which creates more space for intelligent automation of tasks.

In INFRAWEBS SAM is used as a central access point to SWS usage by applications. After an application has started a session with the SAM, it discovers services, selects one of the matching services and executes it within the session.

SAM provides methods for discovery, filtering, ranking, selection, ranking, execution, etc. More detailed examples of the methods and the usage of SAM are provided in the User Manual.

4.8 Semantic Web Services Executor and QoS Broker

The Semantic Web Service Executor (SWS-E) is responsible for executing the Semantic Web Service descriptions based on WSMO. This component handles the interface descriptions of the WSMO Web Service descriptions. Input data is received from the client via the SAM component. This input data is used to evaluate the transition rules within the choreography descriptions. When a rule (or more than one) evaluates to true, an invocation to the underlying service occurs. The result of this invocation is then sent back via SAM. The instance data fed in to the executor is of course at the semantic level, that is, ontology-based. This instance data is also transformed to the equivalent XML data. In a similar way, the XML data which is received after invocation is lifted back to the equivalent ontological level. The executor also makes use of the QoS Monitor during the execution of the services.

The executor is accessed through SAM which is in turn accessed through the application. Consequently, there is not really a graphical interface through which the user can interact directly with this component.

The usage of the SWS-E in INFRAWEBS is hidden by the framework and SAM. A more detailed example about the usage in the case of a testbed is provided in the User Manual.

Summarizing, the SWS-E involves many aspects in semantic web service technology, namely:

- The choreography interface execution involves maintaining a communication state with the client (whether human or automated) of the web service.
- The orchestration component is responsible for communicating with other web services during execution.
- The grounding component is responsible for lifting and lowering the XML and WSML data during communication with real web services such as those exposed via a WSDL interface.

There are different advantages that arise from both the technological and business point of view. This component separates the different aspects during the execution of the web service description. This is also thanks to the inherited WSMO model. Clients of a web service are not interested in how the functionality is achieved but rather in how they can achieve that functionality. The web service itself however needs to communicate with other web services in order to achieve its functionality but this is
hidden away from the client. The use of ontologies as the means of the data model for message exchange brings about interoperability advantages.

On the other hand, the Quality of Service Monitor (QoS Monitor) is responsible for collecting QoS metric data during the execution of the semantic web services and calculates the respective QoS values. These values are then stored in the non-functional properties of the web service and also as instances in the respective QoS ontologies. The QoS Monitor is tightly coupled with the Executor but it can be shipped as a separate jar file. The respective execution engine must then make sure to make the appropriate calls to the monitor in order for it to correctly perform the metric calculations.

The Quality of Service Monitor is responsible for gathering monitor data during the execution of the semantic web services. This data can be accessed either from the local cache of the monitor (obtaining the freshest information) or from the non-functional properties of the specific web service. Such information may be used in different scenarios within the semantic web service life-cycle:

- Ranking for Discovery and/or Selection,
- Design of Composite Web Services, and
- Web Service Replacement.

Once the monitor is started, the usage is simple. Monitor Events have to be created and each one must define the web service identifier, the execution identifier and the status (defined in the ExecutionEvents enumeration). The metric value is automatically updated once the status END, SYSTEM_FAULT or EXECUTION_FAULT have been received. These monitor events are fed to the monitor through the monitor() method.

As with the executor, there is not really an interface to access the QoS Metric data. Rather, these are accessed either through the application or through SAM during the selection process.

The Quality of Service Metrics is used in different stages of the SWS life-cycle. With respect to the INFRAWEBS framework, it can be used within the following contexts:

- During Discovery and Selection;
- During the Design and Composition.

During Discovery and Selection, the discovered web services can be ranked according to some QoS Metric and the user may then choose the web service accordingly. In a similar way during designing and composition, Web services that have adequate QoS values can be chosen.

Though such systems exist in current technological systems, few research efforts have been carried out within the context of semantic web services. Here, we deal with the quality of the semantic web services not of the service itself. This is a very hard measure to obtain and it is very subjective to the clients using the service and the providers selling it.

4.9 Security & Privacy

The security and privacy enabler component operates as an intelligent filtering module attached to an INFRAWEBS application. It is responsible for the validity checking of all information circulating within the application and coming from users and external
services and/or sources. The objective is to maintain the highly dynamic state of the virtual network of services both within acceptable operational bounds and with the maximum possible efficiency and build dynamic trust measures. The adopted approach draws on the biological paradigm of immunity that has grown in popularity within modern IT systems and that defines a highly flexible even if sometimes imperfect "protection shield" around autonomous, selfish individuals to preserve global integrity, autonomy and efficiency. Individual external services are thus treated as somatic components.

From a technical point of view, all circulating messages will contribute to the dynamic development of trust measures for all services or sources involved. This generic setup relies on two assumptions:

- Messages, mostly requests or offers, from services will be processed on a per-message basis and according to integrative functional framework criteria and not under global service authentication criteria.
- Different virtual pathways between external services are in principle created and are competitive with one other, so that continuous self-organization and selection is necessary to come up with a final stable solution to the problem set, again in most cases a complex query for a complex network of services.

This approach also allows to smoothly subsuming static privacy/security policies published by web services - these will be combined with more functional integrity and efficiency criteria. The framework can thus be superposed on external, possibly older, systems.

Apart from the adoption of the immunity perspective and the possibility to subsume other methods, the security shell is lightweight, hence in principle extensible and scalable with minimal development effort. Moreover, it allows proceeding naturally to more advanced conceptual configurations, such as those involving soft, fuzzy and adaptive semantics.

Our approach is situated at the application deployment level within the INFRAWEBS framework. The base security manager is a 3-module system:

- Security monitor. Module responsible for tracking, visualizing and informing about behavioural (functional) changes in the various services involved in an INFRAWEBS application.
- Security reasoner. Module responsible for making sense of those changes, by reasoning on their history. This module also adapts itself to changes, in order to co-develop with the ever-evolving external service environment.
- Privacy monitor. Module responsible for matching, visualizing and informing about user preference profiles in relation to service requirements/policies.

Those high-level tools are encapsulated in a broader customisable manager able to use lower-level INFRAWEBS-independent features and to combine these with the higher-level reasoning modules. This organization will allow to move smoothly to future implementations or extensions of the INFRAWEBS framework that support security features at the system level (such features may include data encryption and message authentication). The same organization also allows using the customisable security manager with other enterprise service buses or organizations, possibly outside the INFRAWEBS context.
5 PROJECT FACTS

5.1 Availability of results

<table>
<thead>
<tr>
<th>Project Result</th>
<th>Responsible Contractor</th>
<th>Available as</th>
<th>Patents or IPR protection</th>
<th>Sector of application</th>
</tr>
</thead>
<tbody>
<tr>
<td>INFRAWEBS Integration Framework middleware (IIF)</td>
<td>ATOS</td>
<td>Rar file Installable</td>
<td>Open source</td>
<td>Cross-sector</td>
</tr>
<tr>
<td>Service Access Middleware (SAM)</td>
<td>SZTAKI</td>
<td>Java and Prolog-based discovery engine</td>
<td>Open source</td>
<td>SWS Researchers</td>
</tr>
<tr>
<td>Distributed Semantic Web Service Repository/Registry (DSWS-R)</td>
<td>SAI</td>
<td>WSMO Studio Plug-in Web service</td>
<td>Open source</td>
<td>SWS providers and researchers</td>
</tr>
<tr>
<td>Semantic Web Services Designer (SWS-D)</td>
<td>IIT</td>
<td>Eclipse Plug-in</td>
<td>Open source</td>
<td>SWS providers WSMO researchers</td>
</tr>
<tr>
<td>Semantic Web Services Composer (SWS-C)</td>
<td>IIT</td>
<td>Web service Eclipse Plug-in</td>
<td>Open source</td>
<td>SWS providers WSMO researchers</td>
</tr>
<tr>
<td>Semantic Web Services Executor (SWS-E)</td>
<td>UIBK</td>
<td>Web service Eclipse Plug-in</td>
<td>Open source</td>
<td>SWS WSMO researchers</td>
</tr>
<tr>
<td>Quality of Service Monitor (QoS Monitor)</td>
<td>UIBK</td>
<td>Web Service Web interface</td>
<td>Open source</td>
<td>SWS WSMO researchers</td>
</tr>
<tr>
<td>Semantic Information Router (SIR)</td>
<td>PROFIUM</td>
<td>Web Service Web interface</td>
<td>Copyright</td>
<td>Horizontal – not limited to any vertical sectors.</td>
</tr>
<tr>
<td>Organizational Memory (OM)</td>
<td>big&amp;Aspa</td>
<td>Web Service Web interface</td>
<td>Copyright</td>
<td>Horizontal – not limited to any vertical sectors.</td>
</tr>
<tr>
<td>Security &amp; Privacy</td>
<td>ICCS</td>
<td>Web Service Web interface</td>
<td>Research Theme</td>
<td>Applied research</td>
</tr>
<tr>
<td>Innovation Monitor®</td>
<td>FTEC</td>
<td>Web Service Web Interface</td>
<td>Copyright and Trademark</td>
<td>Cross-sector</td>
</tr>
</tbody>
</table>

Project results are available for downloading under the respective section Downloads of the project web site:

http://www.infrawebs-eu.org
The INFRAWEBS User Manual is downloadable from:

http://www.infrawebs-eu.org/get_file.php?id=559

This document offers the installation instructions of all components and a guided tour of the whole INFRAWEBS functionality.

### 5.2 Contractors Involved

<table>
<thead>
<tr>
<th>Organization</th>
<th>Contact details</th>
</tr>
</thead>
</table>
| **Fachhochschule Bochum**  
Coordination | Lennershofstraße 140, 44801 Bochum, Germany  
Prof. Dr. Friedbert Pautzke  
friedbert.pautzke@fh-bochum.de |
| **Universität Innsbruck**  
Institute of Computer Science (DERI) | Technikerstraße, 6020 Innsbruck, Austria  
Alice Carpentier  
alice.carpentier@deri.org |
| **Bulgarian Academy of Sciences**  
Institute of Information Technologies | Acad. G. Bonchev St. Bl. 2 and 29-A  
1113 Sofia, Bulgaria  
Prof. Dr. T. Atanasova  
atanasova@iinf.bas.bg |
| **Hungarian Academy of Sciences**  
Department of Distributed Systems | Lagymanyosi u. 11  
DSD H-1111 Budapest XI, Hungary  
Dr. Laszlo Kovacs  
laszlo.kovacs@sztaki.hu |
| **Institute of Communication and Computer Systems** | 15773 Zographou Campus, Athens, Greece  
Dr. Elpida Tzafestas  
brensham@softlab.ece.ntua.gr |
| **Profium** | Lars Sonckin Kaari 12, 02600 Espoo, Finland  
Dr. Janne Saarela  
janne.saarela@profium.com |
| **Ontotext Lab, Sirma Group Corp.** | 135 Tsarigradsko Shosse Blvd., Sofia 1784, Bulgaria  
Atanas Kiryakov  
naso@sirma.bg |
| **FUTUREtec GmbH** | Hauptstraße 188, 51465 Bergisch Gladbach, Germany  
Dr. Caterina Rehm-Berbenni  
crehm@futuretec-gmbh.de |
| **Atos Origin SAE**  
ATOS Research and Innovation | Albarracin 25, 28037 Madrid, Spain  
Tomás Pariente Lobo  
tomas.parientelobo@atosorigin.com |
| **HP Best Innovation Center** | Via Grandi, 4, 20063 Cernusco sul Naviglio, Italy  
Dr. Marco Di Girolamo  
projects.best@hp.com |
| **big7.net GmbH**  
Scientific Coordination | Liebigstraße 13, 40479 Düsseldorf, Germany  
Dr. Hans-Joachim Nern  
nern@infrawebs.info |
### 5.3 Project Characteristics

<table>
<thead>
<tr>
<th>Project Reference No.:</th>
<th>IST – 2003 – 511723</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contract Type:</td>
<td>Specific Targeted Research Project</td>
</tr>
<tr>
<td>Project Acronym:</td>
<td>INFRAWEBS</td>
</tr>
<tr>
<td>Duration:</td>
<td>30 months</td>
</tr>
<tr>
<td>Start Date:</td>
<td>1st of August 2004</td>
</tr>
<tr>
<td>End Date:</td>
<td>31st of January 2007</td>
</tr>
<tr>
<td>Project Cost:</td>
<td>3.065.200 €</td>
</tr>
<tr>
<td>Project Funding:</td>
<td>1.872.000 €</td>
</tr>
<tr>
<td>URL:</td>
<td><a href="http://www.INFRAWEBS.eu">http://www.INFRAWEBS.eu</a> / <a href="http://www.INFRAWEBS.org">http://www.INFRAWEBS.org</a></td>
</tr>
</tbody>
</table>