Towards Semantic Web Service Engineering

Gennady Agre[1], Zlatina Marinova[2], Tomas Pariente[3] and Andras Micsik[4]

[1] Institute of Information Technologies – Bulgarian Academy of Sciences
gennady.agre@abv.bg
[2] OntoText Lab. Sirma Group Corp, Sofia, Bulgaria
zlatina.marinova@sirma.bg
[3] ATOS Origin SAE, Spain
tomas.parientelobo@atosorigin.com
[4] MTA SZTAKI - Hungarian Academy of Sciences
micsik@sztaki.hu

Abstract. The paper presents the main results of the IST FP6 INFRAWEBS project. The project has developed an easy and effective way of constructing and using semantic descriptions for existing and new Web services. INFRAWEBS has adopted the WSMO (Web Service Modeling Ontology) and WSML (Web Service Modeling Language) specifications and has imposed no additional requirements to them. Therefore, the advanced software components developed during the project are of interest to the whole WSMO community. INFRAWEBS offers a SOA framework – INFRAWEBS Integrated Framework (IIF), based on the ESB integration paradigm, which can be easily used by different groups of users (application providers, designers of SWS, etc.). The IIF enables integration of components of different technologies. Furthermore, it can be considered as one of the first frameworks for semantic service engineering that covers the whole SWS life-cycle and allows for creation of complex, semantically-enabled applications.

Keywords: Semantic Web services, Semantic Web Service engineering, Service discovery, Service development, Service composition, WSMO, SOA.

1. Introduction

Semantic Web services (SWS) research is related to automating the development of Web service based applications through semantic Web technology. By providing formal descriptions with well defined semantics, SWS are another step in the direction of solving service engineering problems such as service inter-operation, discovery, choreography and orchestration. There are two major initiatives that work on developing a world-wide standard for the semantic description of Web services: OWL-S [17] and the Web Service Modeling Ontology (WSMO) [20]. WSMO is still under development and has been adopted over the past three years in several Integrated IST Projects such as DIP [9], SEKT [24], Knowledge Web [11] 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 identified in [25]:

- 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,
- The absence of pilot applications focusing on every-day needs of consumers, citizens, industry etc., which can demonstrate the benefits of using semantics.

The IST research project INFRAWEBS[1], 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 on supporting semantic Web service applications within their life-cycle. Being strongly conformant to the current specification of various elements of WSMO (ontologies, goals and semantic Web services), the INFRAWEBS Framework *hides* the complexity of creation of such elements by:

- Identifying different types of actors (users) of SWS Engineering Technologies;
- Clarifying different phases of the Semantic Service Engineering process, and

[1] http://www.infrawebs.eu

- Developing a specialised software toolset, oriented to the identified user types and intended for usage in all phases of the SWS Engineering process.

In the rest of this paper the main features of the INFRAWEBS Framework are presented in more details. Section 2 gives a user-oriented overview of the framework, by identifying the types of INFRAWEBS users, describing the Framework architecture and discussing the support provided by the Framework for its users. Section 3 presents a SOA implementation of the INFRAWEBS Framework. Section 4 presents evaluation results of INFRAWEBS based on the two pilot applications. The last section is a conclusion.

2. INFRAWEBS Framework

Conceptually, the INFRAWEBS Framework is a Service-Oriented Architecture (SOA) comprised of coupled and linked INFRAWEBS semantic Web units (SWU). Each unit provides tools and components (realized as Web services) 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 [8]. The following actors have been identified as potential users of the INFRAWEBS SWS Engineering Framework:

- *Semantic Web Service Provider* – any provider of already existing Web services, who would like to convert them to Semantic Web services and to publish them.
- *Semantic Web Service Broker (Aggregator)* – a provider, who would like to create and publish a service achieving its 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 "ordinary" 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 developed categorization of INFRAWEBS Framework users allowed us 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. The INFRAWEBS Framework is developed to support all stages of the semantic Web service life-cycle (see Fig.1.):

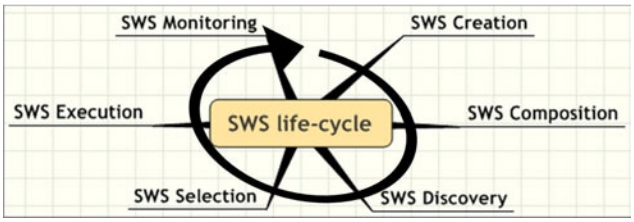

Fig. 1. INFRAWEBS SWS life-cycle

- *SWS Creation* – combines activities related to the creation of semantic descriptions of the Web services, as well as creation of necessary ontologies and goals. Created descriptions are then persistently stored and can be published for common usage.
- *SWS Composition* – already created semantic services can be combined (by their provider, or an external aggregator) to provide new, value-added services. Composed services are also represented and 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 as the WSMO service capability).
- *SWS Selection* – allows users (be they humans 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 invokes it to obtain expected results.

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

The INFRAWEBS framework clusters these stages into two phases: design time and runtime (Fig. 2 and Fig. 3). The components involved in the design time phase are the following:

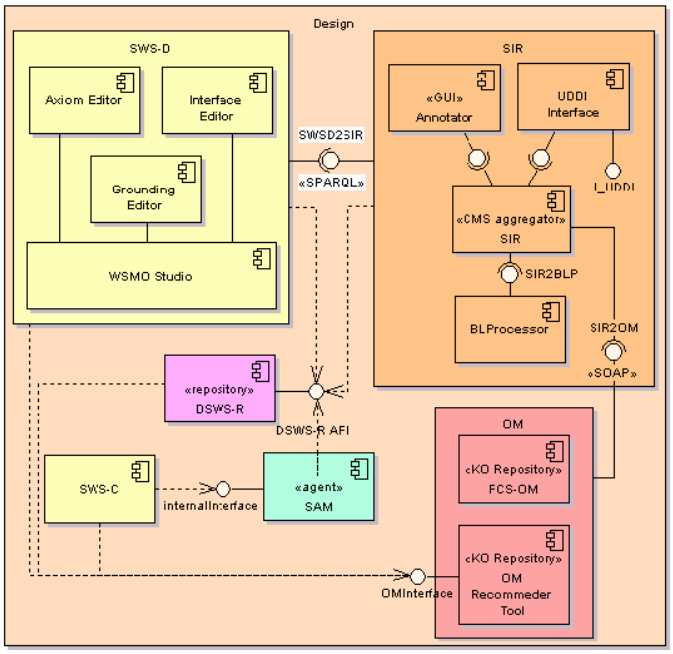

Fig. 2. Design Time Architecture of the INFRAWEBS Framework

- SWS-D (Semantic Web Service Designer) - responsible for the creation of WSMO-based semantic descriptions of Web services and goals
- SWS-C (Semantic Web Service Composer) - responsible for the composition of existing WSMO-based semantic Web services
- DSWS-R (Distributed Semantic Web Service Repository) - responsible for the persistent storage of WSMO-based descriptions and their publication within the Framework.
- SIR (Semantic Information Router) – responsible for registration and annotation of Web services (their WSDL definitions), which are then used in the process of SWS design.
- OM (Organizational Memory) – responsible for indexing and case-based retrieval of WSMO-based descriptions of Web services, goals and ontologies.

The components involved in the Runtime phase are the following:

- SAM (Service Access Middleware) - used by semantic service applications as a middleware supporting the steps of semantic Web service usage, including service discovery, selection and execution.
- SWS-E (Semantic Web Service Executor) - responsible for executing semantic Web services.
- QoS-Monitor (Quality of Service Monitor) - collecting monitored data and calculating quality of service statistics for the semantic Web services being executed.
- DSWS-R – providing run-time access to semantic descriptions of Web services, goals and ontologies.
- OM - used in the first step of the discovery process to retrieve an initial set of semantic Web services matching the current goal, based on ontological keywords similarity.

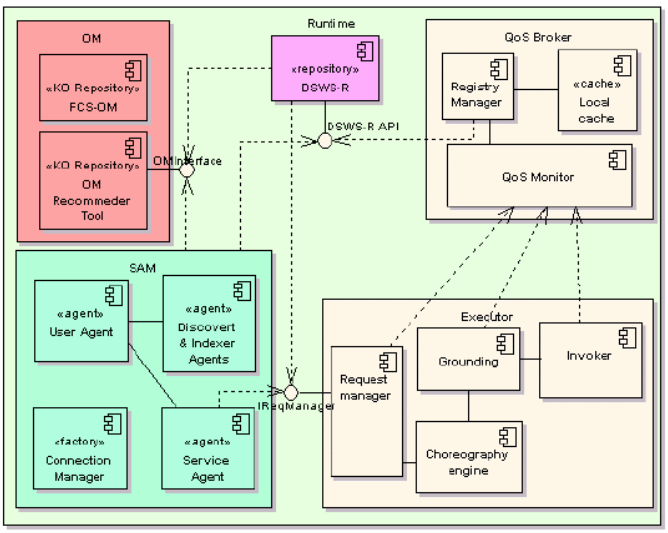

Fig. 3. Run-Rime Time Architecture of the INFRAWEBS Framework

The INFRAWEBS architecture reflects a novel approach to solving problems, occurring in the process of creation of SWS applications, through tight integration of similarity-based and logic-based reasoning. The similarity-based reasoning is used to find fast an approximate solution, which is further clarified by the logic-based reasoning. The following sections show how the architecture described supports different kinds of INFRAWEBS Framework users.

2.1 Use of the INFRAWEBS Framework by Web 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 SWS. The life-cycle of the SWS design process is defined by both the static WSMO-based structure of the semantic description, and by certain assumptions on what kind of additional semantic information is needed, where it is stored and how it can be found. In general, non-semantic Web services are implemented and formally described (as WSDL definitions) outside the INFRAWEBS Framework and are deployed in some Web service container (supported by the service provider).

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 Core[2] schema). The addition of such metadata does not convert a Web service into a WSMO-based SWS - 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 SWS design process and it is seen as a process of registration of Web services into the Framework. The activity is supported by a dedicated component – the Semantic Information Router (SIR), which is a metadata-based content management and aggregation platform for storing and annotating Web services [21]. SIR provides a Web interface for registration, annotation and categorization of WSDL and BPEL files to the INFRAWEBS system.

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:

- Service providers (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 describe semantically.

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 the Simple Protocol And RDF Query Language (SPARQL).

[2] http://dublincore.org/

The next step of the SWS design process is the creation of its WSML description conforming to the WSMO specification. In order to facilitate this very complex activity it is split into several sub-steps:

- Finding a desired non-semantic Web service (described by an annotated WSDL file).
- Finding a set of appropriate ontologies to be used for semantic re-formulation of the main Web service functionality in ontological terms. In the INFRAWEBS Framework ontologies are stored in the WSMO element repositories (DSWS-R).
- Creation of semantic description of the service behavior. According to WSMO, that description consists of service grounding, specifying 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.
- Semantically describing service functionality, advertising what the service can do. Created semantic service capability descriptions are used for service discovery.
- Publication of service descriptions. In the INFRAWEBS Framework the description of SWS (as well as other WSMO-based semantic objects 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, in accordance with the propagation policy specified as part of the advertisement.

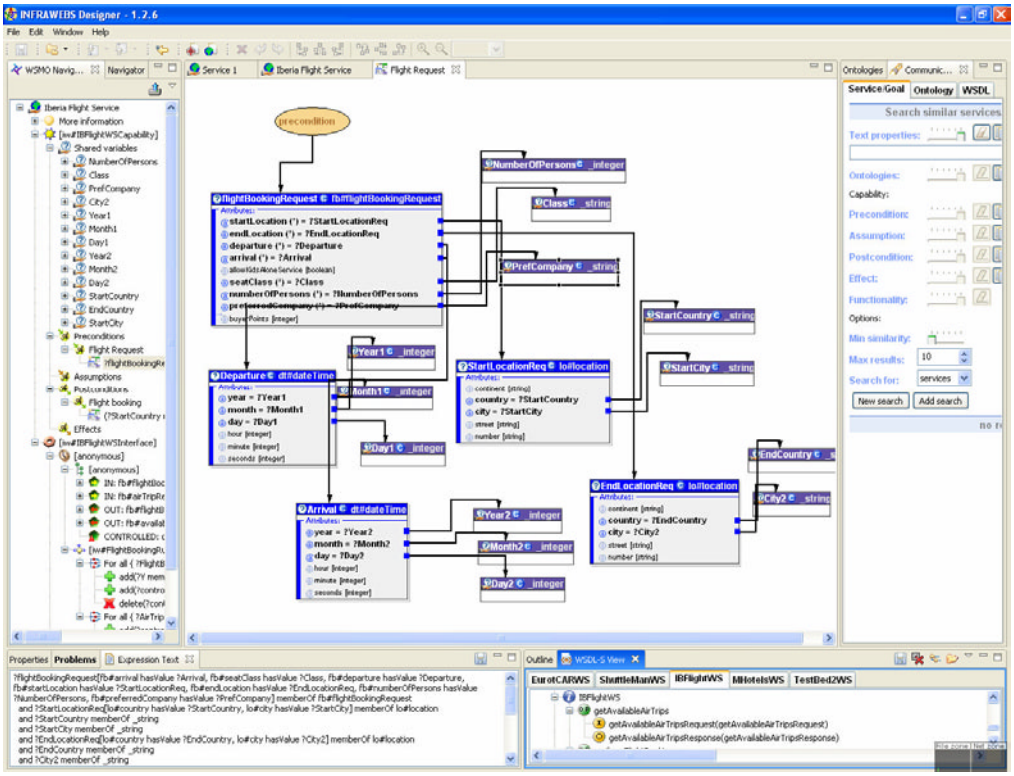

Fig. 4. Screenshot of the INFRAWEBS Designer showing a graphical model of a service precondition.

All of the above steps (except the last one) are performed by means of a special tool – the INFRAWEBS Designer - a graphical, ontology-based, integrated development environment for designing WSMO-based SWS and goals (Fig. 4), [1]. The INFRAWEBS Designer is oriented to Web service providers, and Web service application providers, and does not require any preliminary knowledge of WSML – the logical language used for describing SWS and goals in WSMO [7]. The most important characteristics of the Designer are:

- *User-friendliness:* it proposes an intuitive graphical way for constricting and editing service and goal descriptions, abstracting away as much as possible from the concrete syntax of the logical language used for describing them. The WSML description of the semantic object under construction is automatically generated from the graphical models created by the user.

- *Intensive use of ontologies:* the process of constructing logical descriptions of semantic objects is ontology-driven - in each step of this process the user may select only these elements of the used ontologies that are consistent with the already constructed part of the description.
- *Reusability:* creation of semantic descriptions is a complex and time-consuming process, which can be facilitated by providing the designer with an opportunity to reuse existing, similar descriptions, created by the designer herself or by other users. The INFRAWEBS Designer provides the user with such an opportunity by applying the case-based reasoning approach.

The main novelty of the INFRAWEBS approach is in the combination between the Designer and Organizational Memory (OM), which plays the role of a case-based memory in the INFRAWEBS Framework, [4]. In order to facilitate the process of creating semantic descriptions, the OM can be consulted to find similar semantic descriptions that can serve as templates for the WSMO element under construction (Fig. 5). OM considers all semantic objects as special text documents – "knowledge objects" having specific structure, while each structured part of a document is written in a specific language (natural language and/or WSML). The structure of a document depends on the object type , and in the case of semantic descriptions consists of the main parts representing the element according to the WSMO specification. By means of a special "filtering" procedure (specific for each object type), the OM creates its internal "case" representation of each object - *{T, P, S}*, where *T* is the object type (service, goal, etc.), *P* is the compressed representation of the object content as a structural object feature vector and *S* is the object identifier (IRI), along with some natural language annotation briefly describing the object. In order to provide an efficient filtering process, the OM is equipped with different types of dictionaries – knowledge-domain keywords, generic keywords, abbreviations, stop words and synonyms. It also maintains a list of keyword recommendations that are automatically extracted from each newly filtered object. The OM applies a fuzzy mapping between ontology-based dictionaries and natural language dictionaries, which is used when retrieving similar semantic descriptions by natural language queries.

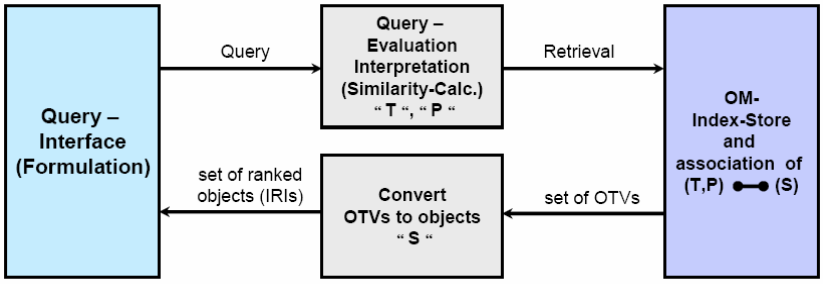

Fig. 5. Scheme for using OM for finding similar WSMO objects

In order to enable the efficient retrieval of the knowledge objects, all their case representations are organized in a hierarchical structure – a multi-layer classification tree, whose internal nodes are centroids and the leaves are concrete cases. Each centroid is a "typical" representative of a set of similar cases, while splitting cases to different sets is done by means of a fuzzy concept matching clustering procedure [16]. Each new case is placed into the classification tree, based on its similarity with the tree centroids. The resulting similarity quotient is calculated using a set of similarity functions, different for each structural part of the case, see [2].

In fact, the use of OM as a knowledge base, containing knowledge about available semantic (services, goals, ontologies) and non-semantic (WSDL files) descriptions, makes the otherwise tedious process of designing a WSMO element (requiring expert knowledge of the WSMO model and the WSML language) a task achievable by ordinary Web service developers.

The main objective for creating a semantic service or other semantic objects (goal, ontology etc.) is to allow this object to be discovered and used by other users. This is achieved by publication of an advertisement of the description in remote registries, provided by the DSWS-R component. Each DSWS-R component provides both functionalities of a registry and a repository, [14]. While the Repository component enables storage and management of WSMO descriptions within the scope of the organization (particular SWU), the Registry is responsible for their publication, propagation and access within the Framework (the network of SWUs). The Registry provides functionality that can be used both to specify which SWS descriptions are publicly available for discovery (and for subsequent composition or execution) and to exchange them with the other registries in the p2p network.

The p2p network is dynamically built by the registries of the SWUs joining the INFRAWEBS framework. Each organization that would like to offer its services (ontologies or goals) and use (or reuse) services provided by other organizations has to:

- Describe its own services as SWS and store them in its local Repository (within the DSWS-R).
- Set up its own registry (part of the DSWS-R) and deploy it on a Web server (as a Web service)
- Add all the endpoints of registries of its business partners in its local Registry configuration, this list of partner registries can be extended at any time.
- Publish advertisements of (some of) its own services in the local Registry and set a propagation policy for each advertisement. Then the Registry will automatically propagate each advertisement, according to its propagation policy, to partner Registries.

When a request to load a particular semantic description (e.g. a SWS) is received by the DSWS-R component it checks whether this description is locally stored or it is advertised in the Registry. Then the DSWS-R automatically retrieves the description either from the local Repository or through a direct request to the corresponding remote Registry.

The DSWS-R functionality can be used by Semantic Service Providers who would like to advertise their services within the INFRAWEBS Framework of partners, so that these descriptions are available for usage and also can be reused by other partners when creating similar descriptions.

2.2 Use of the INFRAWEBS Framework by Web Service Aggregators

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 described as WSMO services, and can then be discovered and used in the same way as atomic services. The INFRAWEBS Design-time Composer tool (SWS-C) enables combining different SWS to obtain a new, complex semantic service, [6]. It uses adapted workflow methodology for creating WSMO-based service compositions, as well as for the determination and visualization of data and control flows within the composed service.

Services to be included in a composition can be selected either directly, by browsing available services, or found through similarity search. In the latter case, a plain-text request is constructed and the Organizational Memory is queried to provide similar services. The result of composition - a WSML description of the service, can be saved locally or persistently stored in the DSWS-R. The graphical schema of the workflow diagram is saved separately as an associated XML file.

Since the WSMO specification of SWS orchestration language is still under development, design-time compositions of services, in INFRAWEBS, are described as choreographies combining the choreography descriptions of participating services. In the future, when the orchestration specification is fully developed, the SWS-C can be extended to create orchestration descriptions on the basis of the defined workflow model.

2.3 Use of the INFRAWEBS Framework by Semantic Service Application Providers

One of the main objectives of the INFRAWEBS Framework is to support organizations that want to create semantically-enriched applications based on semantic Web service technology. In the previous sections the support of the creation of SWS descriptions, via service design or composition, has been explained. The same service creation tools can be used by Semantic Service Application Providers to create the basic functionality of their applications, realised by SWS. However, in order to provide its functionality as services, the application should be able to use all possibilities provided by the runtime INFRAWEBS Environment. At runtime, discovery of a semantic Web service is done via matching a specific user goal description against all service capability descriptions, hence, the first task of the application provider is to prepare a set of goals presenting the application functionality from the user point of view.

According to the WSMO specification [20], each goal is represented as a set of WSML logical expressions and has a structure similar to that of a WSMO service. We consider absolutely unfeasible to assume that either the end-user of a semantic service application will be able to write directly such expressions, or that it is possible to devise an algorithm able to translate automatically each possible natural language query to a corresponding WSML logical expression (goal). That is why, the INFRAWEBS conceptual architecture assumes the presence of a (predefined) set of “general” WSMO-based goals that may be “re-used” or instantiated by the end-users. These general goals (called “goal templates”) should be prepared by the SWS Application provider in design time.

Even though both user goals and goal templates are syntactically represented as WSML goals, the semantic representation of a goal template is slightly different. In order to implement the mechanism for run-time instantiation of goal templates, it is necessary to mark in advance which variables (parameters) of the template are allowed for instantiation (i.e. could be replaced by some concrete values) when formulating a concrete user goal. Such "input" variables are marked in the goal template precondition by means of a special concept with a special attribute ("slots") identifying the variables (as can be seen on Fig. 6).

The assumption that *the functionality of a service application* can be fully described by a *set of goal templates,* which, on the one hand, express all general requests the user can send to the application, and, on the other hand, provide abstractions of all services that can satisfy user goals in this application, is still rather demanding, since it requires the goal templates to be prepared in advance for *all possible* user goals. Being not able to further reduce this demand we, however, are able *to facilitate* the service provider in satisfying it. In order to do this, we assume that:

- Each goal can be represented either by an *atomic* goal template (describing basic functionality of the application) or by *combination (composition) of some atomic goal templates,* and
- The INFRAWEBS Framework provides to the application designer necessary and easy-to-use tools both for goal template creation and for goal composition.

In order to support SWS application providers in creating atomic and composite goal templates, we have developed a special tool - the *Goal Editor*, which is part of the INFRAWEBS Designer. An atomic goal can be created in a graphical way, similarly to the way semantic services are designed, as well as by re-using the descriptions of already existing goals and services. The process of constructing the description of a composite goal is implemented as creation of a new goal, whose capability definition is built by copying the corresponding axioms from several "sub-goals" of the composite goal under construction. Such "copy-paste" way of constructing the composite goal is extended by an additional operation marking "the source" from which the original axiom has been taken (see [3] for further details).

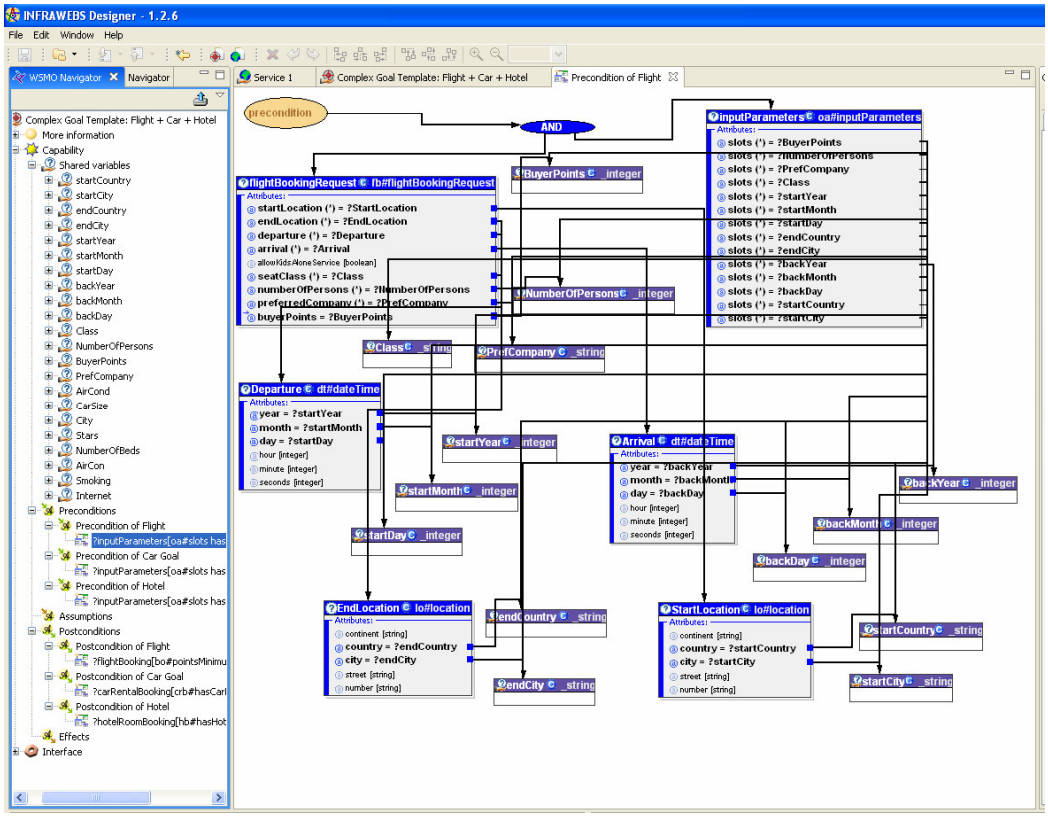

Fig. 6. Screenshot of the INFRAWEBS Designer showing a graphical model of a goal precondition.

The development of two pilot SWS applications (see Section 4) has shown that the integration of the INFRAWEBS Designer with facilities for creating WSMO goals has made it a powerful instrument

for creating atomic and composite goal templates and significantly simplifies the work of the service application providers.

2.4 Running Semantic Service Applications

The functionality of a semantic service application is provided by the semantic Web services used by that application. In order to run such applications, the following functionality is provided by the INFRAWEBS Run-time Environment:

- ***Refinement of goal templates**:* as mentioned in the previous section, semantic service application functionality can be expressed as a set of generic goals (called "goal templates"), describing in terms of WSMO all general requests a user can pose to the application. From this point of view, each particular user request is seen as a concrete instantiation of the appropriate goal template. Thus, creation of a concrete user goal is done by refining a goal template through replacement of its free variables with user-supplied data.
- ***Service discovery:*** the concrete user goal is used for discovering a set of existing semantic services able to satisfy the goal. INFRAWEBS discovery is implemented as a three step process consisting of a pre-filtering step, a step for logical matching and result preparation. The aim of the pre-filtering step is to narrow the list of candidates by using traditional text-processing (keyword matching) algorithms. This step is implemented by sending a request to the OM component to find the most similar semantic services (at the level of ontology keywords representation) to the current goal. It is obvious that the result of such keyword-based discovery may contain non-matches from logical point of view. Nevertheless, the key constraint is to not filter out any good semantic matches in this phase, since logically-incorrect matches will be filtered out by the second step of discovery. During the second, logical step of discovery, the postconditions and effects of the goal and the services are matched to see whether each service can fulfill the request. In addition, the goal preconditions and assumptions are matched against service preconditions and assumptions. To compute semantic matching the INFRAWEBS discovery engine applies the unification facility of Prolog engines. Instead of matching large logical expressions, they are broken into small pieces using Disjunctive Normal Form, where the matching of each piece (clause) can be judged individually. The matching algorithm starts by unification of clauses and variables in both sets representing goal and service. The algorithm iterates over all clauses and labels them as matched, ignored or failed according to the specified unification rules (see [11] for details). The result of discovery is a list of matching services, ranked according to the available simple metrics: number of matching expressions, number of clauses matched in the goal, etc. In order to assist the user in service selection, additional information is provided for each service including service metadata (some of service non-functional properties) and available quality of service data (availability, etc.).
- ***Service selection:*** service selection is done from the list of matching services. In principle it can be done automatically by the application or by the user, on the basis of the annotation information attached to each service in the list. In the INFRAWEBS test bed applications selection is done by the service application user.
- ***Service execution**:* the main objective for running a SWS application is to execute the selected services and present their results to the application user. Generally, service execution is an interactive process carried out between the service requestor and the service provider, based on the defined SWS choreography. In our case the service requestor role is taken by the application which serves as a proxy to the actual user. Thus, the user has no notion of the service choreography, but simply provides (if willing to) data needed for the service to fulfill its task. In the INFRAWEBS Framework service execution is considered as an iterative process of: service invocation that initiates the execution and gets all input data; execution of the selected SWS choreography and providing additional input data if such is requested for the service delivery, see [15].
  - *Service invocation* is done by SAM (Service Access Middleware) based on the identifier of the selected SWS and the description of the user goal, which should be satisfied by the service execution. The description of the goal is automatically transformed into an "input ontology", which is used by the SWS as a source of input data. With this approach, the semantic application needs no knowledge and no special source code for ontology creation, which significantly simplifies the process of application development.
  - *Semantic service execution* is carried out by the SWS-Executor component of the INFRAWEBS Framework, [22], which retrieves the service description and feeds its choreography to the Choreography engine. WSMO choreographies are represented as Abstract State Machines consisting of states and state transitions, [22]. States are defined by the choreography state signature, while transitions are defined through rules. Each transition actually changes the current

state by means of adding, deleting or updating certain facts (instances) as specified by the corresponding rule. During state transitions, the related Web services are executed as defined in the specific SWS grounding. The Choreography engine first initiates the state with the data from the input ontology, and then enables transitions between states by firing the corresponding transition rules. When the current state contains insufficient data and no state transitions can take place, the additional user input may be necessary to continue the execution. Necessary input is specified in the output ontology returned to SAM by the Executor. Execution ends when the Executor sends the final output ontology and does not expect more user input. In case of errors, their types are described in the context object returned to SAM, which transfers it to the application,

- When *additional input data* is needed from the user, SAM sends to the application the current output ontology and the type of the variables that have no value. Instances in the output ontology are converted into the corresponding object structures that can be processed as plain Java objects. In that way, the application can easily find the empty attributes of the missing concept and fill them with relevant values. Such an approach demands considerably less code and expertise of semantic technologies from the SWS application provider.

2. 5 Run-time Service Composition

The INFRAWEBS user request modeling enables definition of "complex goals" which define independent user objectives that are interconnected by certain constraints. A typical example of a complex goal in the SFS scenario (see Section 4.1) is a request to book a flight for certain dates and reserve a room at a hotel in the flight destination city for the same dates. In this example, the independent goals are booking a flight and making hotel room reservation, but they are implicitly constrained by the same arrival and check-in dates as well as by same flight destination and hotel location. Complex goals are also predefined by application providers at design-time and capture any possible combination of "simple" goals that can be useful for the application functionality.

When the application user works with functionality represented as a complex goal (e.g. creates a user package for flight reservation from Munich to Madrid on March, 8th and books a hotel in Madrid for the same date), the same mechanisms for goal instantiation and discovery are used. In this case the discovery can find either atomic or composed services that match the goal instance, but in some cases no single service will match the goal. In such cases the INFRAWEBS run-time composition mechanism ([3]) is used to dynamically find an appropriate solution.

The Run-Time Composer performs its work in two phases. In the first phase (Fig. 7) the user goal is recursively decomposed into two sub-goals, one of them is atomic and the other one is either atomic or composite. A goal is always decomposed back to the original sub-goals that compose it. This is done using the "source" information kept as part of the composite goal description, see [3]. Both sub-goals are matched to services. If the composite sub-goal does not match any services, then further attempt at its decomposition is made. If no matches for an atomic sub-goal are found at some point, the process of run-time composition fails. When matching services are found for a sub-goal the application user is asked to select one particular service to be included in the ad-hoc composition set. The rest of the matching services are also cached as alternatives that can be used for substitution in case of execution failure.

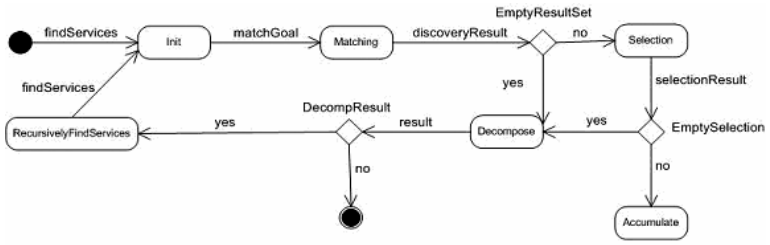

Fig. 7. Service Composition Finder Logic

The second phase of run-time composition is composition execution, when services of the dynamically composed set are iteratively executed, one by one (Fig. 8). Each service execution may result in providing the expected outcome (successful execution), in failure or in user input request, as described earlier. Composition context (input and output ontologies) accumulates all outcomes (instances) generated by successfully executed services.

If a service requests additional input, its execution is stopped and different options for getting this input are investigated. If any of the other services in the run-time composition can provide that input as a result of its execution, the service is put in a waiting list and execution continues with the next service in the composition. If no service can provide this input, the user is asked to enter it through the application. When all services of the composition have already been executed, the services in the waiting list are executed again based on the continuously updated context. The process is iteratively repeated until all services in the dynamically composed set are executed successfully or one of them fails. If any of the composed service executions fails, service replacements are presented to the user in order to continue the execution with an alternative service.

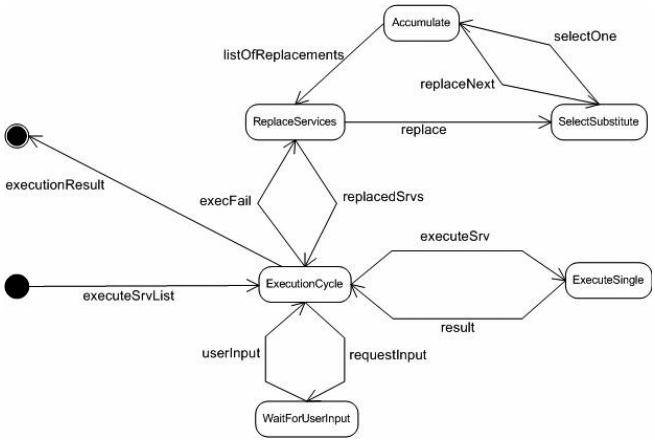

Fig. 8. Composition Execution Engine Logic

3. Integrated INFRAWEBS Framework

The INFRAWEBS Framework is implemented as an extensible Enterprise Service Bus (ESB) middleware that exposes the public methods of the INFRAWEBS components and can be easily extended by external components or services, [12]. The *Integrated INFRAWEBS Framework (IIF)* can be seen as an underlying infrastructure for communication and integration of all the INFRAWEBS components, and as a unique selling point for exposing the functionality of such components to the external world in the form of services. The IIF is deployed in a peer-to-peer network (Fig.9), with possible integration of components with different technologies within the peer. The IIF allows access to the methods of the components via Java APIs or Web service wrappers, so any application able to use these technologies can interoperate with INFRAWEBS components. IIF provides native support for Java-based components and partial support for non-Java based components, using WS technology.

Every peer can be deployed containing all or part of the set of INFRAWEBS stack of components. The IIF hides from the INFRAWEBS users the complexity of dealing with this p2p architecture.

The IIF provides an *IIF Connector* instantiated by Java-based components to allow them to exchange messages and an *IIF Server* routing those messages to the appropriate recipient. The *IIF Server* exposes a complete Web service implementation, which collects the full set of operations supported by the components plugged within the IIF. The service can be used by non-Java components to send messages to other components plugged within the IIF infrastructure or to external applications. IIF provides two ways of communication:

- *Specific* - by instantiating a concrete *IIF delegate* for a particular component's interface to which a message should be sent. The delegate "knows" the requested component and details on how to build and send the message.
- *Generic* - IIF provides methods to generically invoke a remote method. In this case the user just has to know the name of the component method and its parameters.

For both ways of communication IIF supports different kinds of message exchange patterns such as one way, in/out synchronous, in/out asynchronous, publication-subscription and broadcasting (see [12] for details)..

The INFRAWEBS *IIF 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 creation, maintenance and execution of WSMO-based SWS, it can be considered as a Semantic SOA. The use of the underlying ESB middleware provides the IIF with the necessary extensibility to be a Semantic ESB.

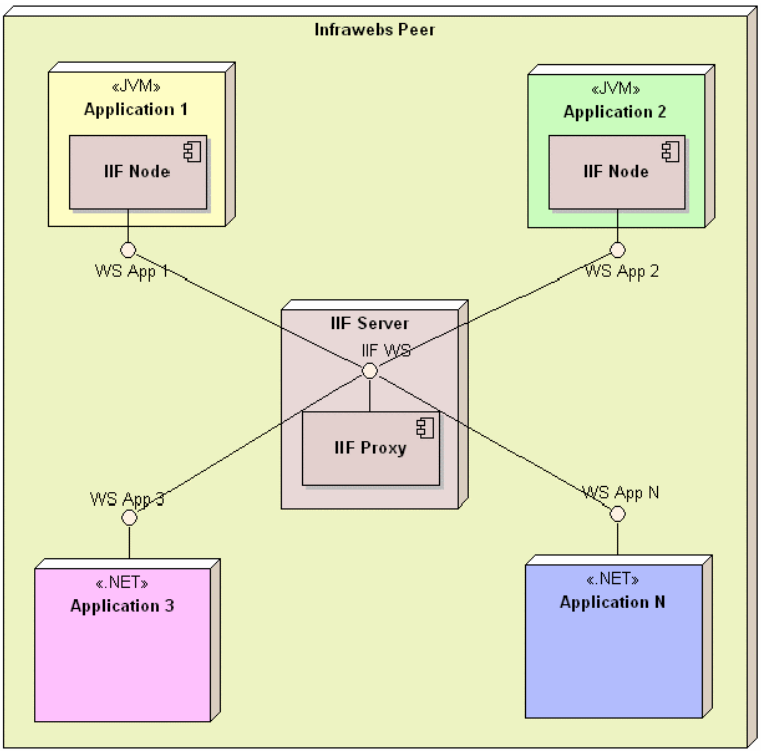

Fig. 9. IIF peer-to-peer architecture

4. INFRAWEBS Framework Evaluation

The INFRAWEBS Framework has been evaluated in two test beds – the first is used in a travel agency scenario [13] and the second is based on an eGovernment scenario [19].

4.1 Test bed 1 - Stream Flows! System

STREAM Flows! System (SFS) is a first prototype application of the IIF, which aims at overcoming such shortcomings of existing Frequent Flyers Programs as impossibility of their users to contract services, or combination of services, using asynchronous, real-time, anywhere and anytime system. The owner of the SFS is STREAM Airlines. Users can obtain points or miles from purchasing services of the STREAM group. These services can be airline tickets, hotel bookings, car rentals and many others. The customer purchases services (paying for them in any kind of money transfer) from many companies (engaged with the SFS program), which collect the information of the customer and send it to the SFS, adding the counterpart of the service in points to the SFS loyalty programme. The SFS semantic Web application collects all the information and stores it into its own databases.

The SFS uses the framework supplied by INFRAWEBS both for design and for runtime activities. First of all the Service Providers have access to the Designer (SWS-D) and the Composer (SWS-C), in order to define the SWS descriptions and compositions that are needed. In general, SWS descriptions can be provided by other parties, and be stored and advertised in different repositories of the INFRAWEBS p2p network.

The main functionality of the SFS allows the user to create or select travel packages. The user is able to create new packages using the framework provided by INFRAWEBS. The choice of a complex

package triggers the selection of an appropriate composite goal template allowing dynamic composition of services (e.g. flight + hotel + car rental) during execution. SFS semantic application was described by 12 ontologies, 11 semantic Web services and 16 goal templates.

The test bed has shown several benefits of using applications based on SWS:

- Run-time composition of services: the use of composite goal templates enables easy creation of basic compositions of services in runtime.
- Selection of best offers based on SWS discovery: the use of semantics allows for finding the most appropriate services matching the user's needs expressed as a goal.
- Service providers have fewer difficulties adding new services: each new service that can be potentially used by the application should simply be described as a SWS and incorporated to the repositories, thus minimizing the integration process.
- Ideally no modifications to the code are required: the user input data forms can be created dynamically based on goal template slots (inputs), minimizing the impact on the maintenance of the application.
- As a J2EE application SFS system has been easily integrated with all INFRAWEBS components by using the Java API and the connectors provided by the IIF.

4.2 Test bed 2 – Opening New Business

The second test bed implements an e-Government scenario related to the process of opening a new commercial activity (for example, adding a new shop to an already existing shop chain). Usually, a new shop opening is bound to current local laws and regulations that may be changed in time. That is why, the user should first be familiar with the list of needed documents and certificates required to open the new activity. Then the user has to ask for each document the office or agency authorized to release such a certificate. The process is time consuming, as users need to travel to each appointed agency to post the request, sometimes return back to get the certificate if not shipped to them. The user must also know the proper relations or dependencies among released service certificates.

The proposed SWS application allows any user to take advantage of new technology for accomplishing all needed activities from her personal computer. It enables huge saving of time, error free processes, and much faster responses. The test bed was presented by 9 ontologies, 4 semantic Web services and 4 goal templates.

The test bed offers interesting aspects that complements the results of the SFS test bed:

- The application is well ahead with current services offered to customers.
- The application is a .NET (C#) application that uses the IIF Web service implementation. This has proved the possibility of integration of multiples technologies provided by the SOA approach followed by the IIF.
- Although it is a simple demonstrator aimed at taking advantage mostly from the semantic content, the application could be used by government agencies to show the main advantages they could offer to their citizens and how to dramatically decrease the internal costs due to automatic interaction and less personnel required.

4.3 Framework Evaluation Results

INFRAWEBS Framework has been evaluated in two dimensions ([18]):

- *Environment dimension*: measuring the degree of satisfaction and fulfillment of the Framework objectives with regards to both test bed results, based on the opinion of the different IIF users – SWS providers, SWS test-bed application providers and application end-users.
- *System dimension:* this evaluation takes into account the development of the overall service provision chain (annotation, design, composition, execution, monitoring), focusing on the primary (innovation–related) questions that have been identified. The system dimension does not depend on a particular demonstrator, because it is more related to the Framework capabilities and features.

The evaluation has shown that INFRAWEBS offers a framework covering the whole SWS life-cycle - from design and static composition - to discovery, run-time composition, execution and monitoring. It gives the possibility of creating a new set of semantic-enabled applications that was not possible to develop before in such a consistent way with pre-existing platforms or frameworks.

Moreover, INFRAWEBS is fully based on the current state of WSMO and provides advanced tools for the whole WSMO community. The INFRAWEBS design-time components offer graphical user

interface for designing and publishing the semantic descriptions of WSMO-based Web services and goals in an easier way than before. They allow the user to create such semantic objects without any knowledge of WSML language used for their descriptions.

All INFRAWEBS components make use of the INFRAWEBS Integration Framework (IIF) to communicate with each other. Such Framework allows the integration of components of different technologies, and by using Web service interface can be used as an open framework. The use of both a Java-based API and Web service interfaces for the IIF methods allows the application providers to access INFRAWEBS features independently of the technology they use. Moreover, a unified API, provided by the SAM component, allows application providers to interact with the run-time INFRAWEBS Environment without writing or reading any WSML expressions, significantly facilitating the process of creating semantically-enabled applications.

Along with the clear benefits of INFRAWEBS, the evaluation process has allowed to identify some underlying assumptions that should be known for better understanding the applicability of the INFRAWEBS Framework and trends for its future development. The most important of them are:

- All INFRAWEBS components have been developed in strict conformance with the current WSMO specification. Since the WSMO itself is in the process of active development, it can lead to necessity for adjusting the future version of the INFRAWEBS Framework.
- The current version of the INFRAWEBS Framework does not support ontology mediation. All ontologies used are assumed to be known by all service and service application providers. The ontologies are shared using the DSWS-R functionality.
- User requests to a SWS application are expressed as WSMO goals, which are automatically constructed based on the predefined goal templates created in advance by the application developers. This restricts a set of possible goals that potentially can be formulated by the user, but allows the users with no knowledge of WSMO to use INFRAWEBS applications.
- The INFRAWEBS run-time composition is only possible for goals created from goal templates preliminary designed by the INFRAWEBS Goal Editor. That is why, in order to be decomposable, a WSMO goal created by any other tool should be rewritten by means of the Goal Editor.
- Since INFRAWEBS deals with WSMO-based semantic services, the execution of such services requires the presence of so called “adapters”, which ground ontological concepts used in the semantic descriptions to the XML Schema data types used in the WSDL descriptions of the Web services. Automatic creation of such adapters is an ongoing research in WSMO, and that is why, the current version of the INFRAWEBS Framework does not include tools for adapter design.
- Because of lack of clear specification of WSMO-based service orchestration, the current version of INFRAWEBS Composer offers a limited support for orchestration - only on the functional level.
- The INFRAWEBS choreography engine does not clearly guide the user during the execution process. A better engine would be a desirable future enhancement.

5. Conclusion and Future Trends

In this paper we have presented the main results of the IST FP6 INFRAWEBS project. The project has developed an easy and effective way of constructing and using semantic descriptions for existing and new Web services. The advantage of semantic Web services is that they can be understood better by other programs and human beings, since not only syntactic information is available, but also the meaning (semantics). The presence of such semantic information allows for better possibilities for searching and finding useful services.

The project has adopted the WSMO and WSML standards and imposed no additional requirements to them. Therefore, the advanced software components developed during the project are of interest to the whole WSMO community. Most of the developed components are available as open source software under LGPL license.

One of the novel aspects in the approach of INFRAWEBS is the combination between the Organizational Memory (OM) and the Semantic Web Service Designer. This tool allows a user to compose a WSMO-based semantic description of a given Web service based on existing WSDL description of this service and a set of WSML ontologies. To ease this process the OM can be consulted to find look-alike semantic Web services that can serve as templates for the WSMO object under construction. In fact, the use of the OM as a knowledge base holding knowledge about available semantic Web services makes the otherwise tedious process of composing a WSMO object (requiring expert knowledge of the WSMO model and WSML language) a task doable for the ordinary web

service developers. Easing the hurdle of WSMO object construction is an achievement of the project that has a potential impact on the adoption of semantic Web services on a larger scale.

INFRAWEBS offers a SOA framework (IIF) based on an ESB middleware which is easy to use by different users (application providers, designers of SWS, etc.) and allows the integration of components of different technologies.

All of the above mentioned analysis allows us to conclude that INFRAWEBS is one of the first frameworks for semantic Web service engineering that covers the whole SWS life-cycle and allows creation of complex semantically-enabled applications.

References

1. G. Agre. INFRAWEBS Designer – A Graphical Tool for Designing Semantic Web Services. In: *AIMSA 2006, LNAI 4183*, Springer-Verlag Berlin Heidelberg, 2006., 275-289.
2. G. Agre. Using Case-based Reasoning for Creating Semantic Web Services: an INFRAWEBS Approach. In: *Proc. of EUROMEDIA'2006*, April 17-19, 2006, Athens, Greece, 130-137.
3. G. Agre and Z. Marinova. An INFRAWEBS Approach to Dynamic Composition of Semantic Web Services. *Cybernetics and Information Technologies (CIT), Volume 7, №1*, 2007, Bulgarian Academy of Sciences, Sofia, 45-61.
4. G. Andonova, G. Agre, H.-Joachim Nern, A. Boyanov. Fuzzy Concept Set Based Organizational Memory as a Quasi Non-Semantic Component within the INFRAWEBS Framework. In: Bernadette Bouchon-Meunier (Ed.) *Proceedings of IPMU2006*, July 2-7, 2006, Paris, France, Editions EDK, ISBN 2-84254-112-X, 2268-2275.
5. http://asg-platform.org/, Last accessed May 2007.
6. T. Atanasova, H. Daskalova, V. Grigorova and D. Gulev. Design& Realisation of Case-based Composition of SW services in Design Time (Design-time Composer). *INFRAWEBS Deliverable D5.4.2*, September 2006.
7. Bruijn, J., Lausen, H., Krummenacher, R., Polleres, A.; Predoiu, L., Kifer, M. and Fensel, D. D16.1 – The Web Services Modeling Language (WSML). *WSML Draft*, October 2005.
8. M. Colombo, E. Di Nitto, M. Di Penta, D. Distante, and M. Zuccal. Speaking a Common Language: A Conceptual Model for Describing Service-Oriented Systems. In: *Proc. of the 3rd International Conference on Service Oriented Computing (ICSOC),* Amsterdam, the Netherlands, December 2005.
9. http://dip.semanticweb.org/, Last accessed May 2007.
10. L.Kovács, A. Micsik and P. Pallinger. Handling User Preferences and Added Value in Discovery of Semantic Web Services. In: Proceedings of the IEEE International Conference on Web Services (ICWS 2007), 09-13 July 2007, Salt Lake City, Utah, USA, pp. 225-232,
11. http://knowledgeweb.semanticweb.org/, Last accessed May 2007.
12. A. López, T. Pariente and Y. Gorroñogoitia . Final INFRAWEBS Integrated Framework (IIF). *INFRAWEBS Deliverable D11.3.6*. February 2007.
13. O. López, A.López Pérez, C. Pezuela, Y.Gorroñogoitia, E. Riceputi. Requirement-Profiles & Technical Risk Management & Demonstration & Testbeds & Specification Checker. *INFRAWEBS Deliverable D10.1-2-3-4.1*, November 2005.
14. Z. Marinova, Agre, G., Ognyanov, D. Final Dynamic DSWS-R and integration in the IIF. *INFRAWEBS Deliverable D4.4.3*, February 2007.
15. A. Micsik, L. Kovács, P. Pallinger, Z. Tóth. Specification and Realisation of the Execution Control Component. *INFRAWEBS Deliverable D6.3.3*, February 2007.
16. J. Nern, A. Boyanov and G. Jesdinsky. Specification & Realised Reduced Organisational Memory and Recommender Tool. *INFRAWEBS Deliverable D2.1.1*. November 2005.
17. http://www.daml.org/services/owl-s/, Last accessed May 2007.
18. T. Pariente Lobo, A. Lopez Perez and J. Arnaiz Paradinaz. Demonstrator. *INFRAWEBS Deliverable D10.8.3*, February 2007.
19. E. Riceputi, Requirement Profile 2 & Knowledge Objects. *INFRAWEBS* Deliverable D10.5-6-7.2, September 2006.
20. D. Roman, U. Keller, H. Lausen (eds.) Web Service Modeling Ontology, October 2006, *WSMO Final Draft*.
21. J. Saarela. Specification & General SIR, Full Model & Coupling to SWS Design activity. INFRAWEBS *Deliverable D3.2-3-4.2*, July 2006.
22. J. Scicluna, A. Polleres, Dumitru Roman and D. Fensel. D14v0.2. Ontology-based Choreography and Orchestration of WSMO Services. *WSMO Final Draft* 3rd February 2006.
23. J. Scicluna, Marinova, Z.; Agre, G. D7.4.3 Final SWS-E and Running P2P-Agent. INFRAWEBS Deliverable *D7.4.3*, February 2007.
24. http://sekt.semanticweb.org/, Last accessed May 2007.
25. Software, Services and Complexity Research in the IST Programme: An overview. *Software Technologies Unit, Information Society and Media DG, European Commission, Ver. 1.0, draft*, September 2006.