19

Implementing Innovative IT Solutions with Semantic Web Technologies

Vili Podgorelec and Boštjan Grašič
University of Maribor
Slovenia

1. Introduction

With modern transportation, communication, and business connections, distances are becoming narrower and competition tougher. Therefore, successful companies nowadays need to adapt to changes in environment more rapidly than they used to. Besides ever rapidly changing environment, an organizational shift towards customer has been noticed. For the last ten years or so there has been a steady international move towards changing the way customer services are delivered, financed and regulated, with the main purpose being the improvement of efficiency so that more customers could receive better service more quickly without reducing (and possibly increasing) the quality.
On the other hand, the world is witnessing a remarkable proliferation of new knowledge. With the development of information technology the amount of various business data being created and stored is growing exponentially. The endeavour of researchers and engineers to take advantage of this new knowledge, coupled with an increasing need to use limited resources more efficiently, has presented unique challenges. As new knowledge and solutions to systemic problems is sought for, it is appropriate to ask how the revolution in information and communications technologies may facilitate efficiency and prevent unnecessary duplication of effort.
Many times it has been proven that the proper use of proper knowledge is the best way of optimizing work processes. In order to improve the usability and performance the effort duplication should be minimized, access to information resources improved and possibly unified, and the information technology utilized in such a manner that would allow users to make advantage of information they need without having to bother with the abundance of them.
As the evolution of information technology and software design progresses the possible solutions to the above idea could be knowledge management and web-based software services, combined within a unified technology. Based on our experiences in developing software solutions, it is our belief that semantic web technologies could be the one technology to solve this task. In the following sections we will try to present the properties of this technology together with some advices on how to utilize it properly.

2. Semantic web technologies

The idea behind semantic web is fairly simple. Main idea is that computers would be able to understand the meaning of the data. Passin defines semantic web as a vision or a liquid,

developing and informally defined concept (Passin, 2004). According to Passin, vision of semantic web is that computers would be able to find, read and understand the meaning of data. Tim Berners-Lee, sees semantic web as "web of data" compared to web of documents as we know world wide web today (Berners-Lee et al., 2001).

There exist many scenarios of semantic web usage. Most of them include autonomous agents that are able to autonomously find data on the web of data and present information that is relevant to us. If we look at the situation from today's point of view, one would have to search the search engines about the key-word of the subject of interest, then he would have to search the resulting web pages for information, he is interested in.

Technologically speaking we are still far from the discussed scenario, but there already exist technologies that should enable machines to operate with data and its meaning. These technologies are called semantic web technologies (SWT). There were many attempts to build core SWT – according to Passin, the most promising are technologies that are supervised by W3C consortium (Passin, 2004). These technologies are mostly built upon technologies from the Defense Advanced Research Projects Agency (DARPA) and their DAML (DARPA Agent Markup Language) language. SWT are based on XML language that enables them to be platform and program language independent. SWT are extensible like XML. This means that for each purpose there exists a separate technology that is compatible with others.

SWT are built in layers, as shown on Figure 1. Each upper layer provides additional functional aspect and is based on the lower one, with which is fully compatible. Most bottom layer includes Unicode and URI technologies. Unicode enables SWT to be platform and language independent. URI is not just used for electronic resource identification, but for general resource identification. Each resource, also a physical one, is in the semantic web represented with an URI descriptor.

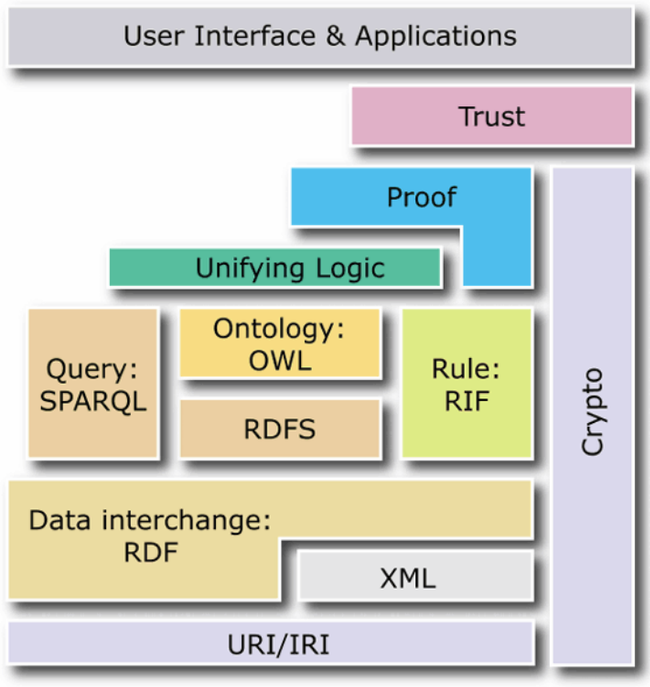


Fig. 1. The layers of semantic web technologies

The second layer of enabling technologies for SWT includes XML, XML Schema and namespaces. Semantic web data is represented in XML language, this implicates that semantic web documents are actually XML documents with predefined schema.

2.1 Describing resources within semantic web technologies: RDF and ontologies

Above the enabling technologies with which we can get only self describable documents is RDF layer. RDF is core SWT and is acronym for resource description framework. It is used for describing resources of any kind. Resources are represented with an URI descriptor. RDF document is a composition of statements called triples. Each triple is a statement about a resource and is composed of subject, predicate and object. While subjects and predicates are URI resources, an object can be either an URI resource or a plain string called literal. Literals cannot be identified; therefore they cannot be referenced and used as objects or predicates (Lassila & Swick, 1999).

Because of RDF's flexibility and its simple structure, it can be used to describe practically any resource. Because of its structure, an RDF document can be represented as a directed graph, where subjects and objects are represented as nodes, while predicates are represented as edges.

Above the RDF layer in SWT stack is ontology. Ontology is a rigorous and exhaustive organization of some knowledge domain that is usually hierarchical and contains all the relevant entities and their relations (Wordnet[1]). For ontology description there exist more languages, most common nowadays is OWL (Web Ontology Language), which is derived from DAML and OIL. OWL is vocabulary extension of RDF, which is why an OWL document is also a valid RDF document. Rules and characteristics of RDF also apply to OWL (Passin, 2004).

OWL uses description logic to classify resources defined in a RDF document. OWL 1.1 is divided in three subsets: OWL Full, OWL DL and OWL Lite. Instead of using these sublanguages, OWL 2 introduces profiles (OWL 2 EL, OWL 2 QL, OWL 2 RL). Difference between different sublanguages is in number of language constructs they support. Different sub-sets were developed because of compensation between performance and expressiveness. OWL Lite has the best performance and limited expressiveness, while OWL Full has opposite characteristics. DL in OWL DL stands for description logic. It was designed for existing description logic business segment and should be preferred over OWL Full in production environments. Main advantage over the full subset is that OWL DL is fully decidable (Dean & Schreiber, 2004).

We already mentioned that OWL is used for classification purposes. It has two main categories: classes and properties. RDF nodes (subjects and objects) are categorized in OWL classes, while RDF predicates are categorized by OWL properties. Nodes and predicates are being classified using description logic predicates. Classes in OWL are treated as sets and OWL supports all set operations (union, intersection, disjoint, equivalent, subset) (Dean & Schreiber, 2004).

Next layer which is also important in SWT stack is logic layer. Rule based languages are natural choice for this purpose. W3C consortium has released a member submission called SWRL (Semantic Web Rule Language). SWRL is rule-based language that is combination of OWL Lite and OWL DL subsets with Datalog RuleML sublanguages. With SWRL language, knowledge engineer can define rules that are then applied to OWL knowledge database.

[1] http://wordnet.princeton.edu

Rules are composed of antecedent and consequent. If the requirement defined in antecedent is met then the triple in consequent is inserted in the graph. The two top most technologies (trust and proof) in SWT stack are not supported yet. Because there is no technical solution to these problem areas available we will ignore these two layers.

2.2 The key role of ontology

What is the purpose of ontology in semantic web? Ontology describes the subject domain using notions of concepts, instances, attributes, relations and axioms. Ontology can be defined as a formal explicit specification of a shared conceptualization. It is a useful way to organize and share information while offering intelligent means for knowledge management. Ontology also enhances semantic search in distributed and heterogeneous information services. Ontologies are the key player, if we want to do (automatic) search in more advanced ways, not only keyword search.

There are several benefits of using ontologies for information solutions. Semantic search engines return instances that constitute answers to queries rather than documents containing search strings as in keyword search engines. Semantic search uses meanings (semantics) of the query terms defined in the ontology. The data of ontology constitutes precise answers to user questions. Users can further browse related concept because answers are interconnected through semantics. It can be speculated that using ontology supported systems users will also be able to invoke functionalities or query data using free text input in the future.

The central part of a semantic web application is an ontology that describes some knowledge domain using notions of concepts, instances, attributes, relations and axioms. It is a useful way to organize and share information while offering means for enhanced semantic search in distributed and heterogeneous information systems. Ontology can be defined as a formal explicit specification of a shared conceptualization.

Nowadays, there are many application domains, where the utility of ontologies is widely accepted, and where ontologies have already been deployed at large scale. However available ontologies, actually used as common vocabulary for certain applications, cover particular domains to different granularities and cannot be directly used for the semantic web. This issue has been addressed for example in the medical domain in project GALEN[2], where the authors developed a special representation language, tailored for the particularities of the (English) medical vocabulary. However, the usage of a proprietary representation makes the ontological knowledge difficult to be extended by third parties or in a semantic web setting.

In order to adopt the SWT for supporting the information management in a specific domain, there is a key field that needs to be addressed: domain knowledge that we want to integrate within the domain. For this purpose an ontology needs to be defined, which will then allow all further actions, like semantic annotation of data (in accordance with the ontology), integration of data resources, advanced searching and inferring on the data.

It should be pointed out, that there are two notions of the term ontology: heavyweight and lightweight ontology. Heavyweight ontologies are mainly used for complex, logic based modeling of a knowledge intensive domain (e.g. gene ontology, protein ontology). They are carefully designed throughout a strenuous and vigorous process involving a consortium of

[2] http://www.opengalen.org

experts; the specification of concepts, relations and logical restrictions is very precise. On the other hand, lightweight ontologies are used mainly for data integration purposes or they act as a common vocabulary; in this way concepts may be defined more loosely and are practical goal oriented. In this paper we are using the term ontology as a lightweight ontology used for integrating information resources.
The integration of information (or knowledge) within a specific domain has always been a significant issue. The introduction of ontologies and SWT have provided some promises in this direction. Some knowledge integration approaches using SWT have already shown some results, either as an integrative mechanism within an organization (Lenz et al., 2007) or inter-organizationally (Knaup et al., 2007).

3. Common SWT based system architecture

As already said, SWT are usually represented in a layer cake model (Figure 1), where XML and URI are foundations for SWT building blocks. First SWT specific building block is RDF (Manola & Miller, 2004) – a language for describing resources, which are represented as URI-s. Next important building block is OWL (Dean & Schreiber, 2004) – a language for describing ontologies. Next to ontology building block is RIF, which stands for rule interchange format and includes rule languages that are compatible to RIF. One such rule language widely used in SWT is SWRL – semantic web rule language. It enables reasoning on concepts defined in OWL ontologies.
A typical SWT system is based upon RDF, OWL and a rule language compatible to RIF (SWRL is widely used). In this manner, RDF is mainly considered as a data backend and a data interchange technology. Concepts that are defined in ontology are mostly used for data integration. Rules enable encapsulation of business logic; based on defined rules, new knowledge is being inferred according to concepts defined in ontology and RDF data. A query language (like SparQL) is used to query the semantic data. These building blocks represent the core SWT. A typical SWT based system architecture is shown in Figure 2.

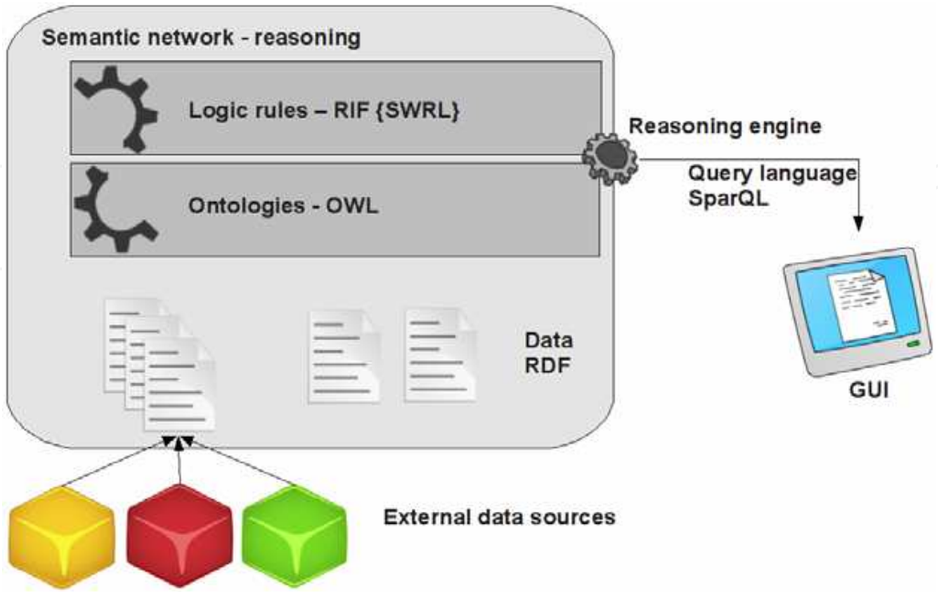


Fig. 2. Common SWT based system architecture

SWT provide means for knowledge representation. Opposite to past research efforts in the field of artificial intelligence, where knowledge representation systems were mostly centralized and isolated, SWT are designed to be used on a Web scale in heterogeneous information environment. While SWT provide fairly expressive formalisms for knowledge representation and reasoning, this often results in poor performance.
To overcome this issue, we propose division of information space represented in semantic network into two subsets:

- static information – information that is essential for reasoning purposes, hence has to be always available to the reasoner,
- dynamic information – information that is not needed for reasoning purposes, thus it can be fetched when needed.

By dividing information space into two subsets and excluding one from semantic network, the size of the semantic network is reduced; hence the amount of information that needs to be processed by the reasoner is reduced. For providing dynamic information, we propose use of semantic web services that are executed automatically, when the demand for dynamic information arises.
This way not only the performance is increased, but the interoperability with legacy systems is increased also. Data provided by services is not limited to data that is excluded because of performance issues. Based on service oriented architecture paradigm, services can also act as information providers from legacy systems in loose coupling, higher reuse and greater interoperability manner.
Having in mind two mayor drawbacks of a common SWT based architecture, namely poor performance on large data sets and high data integration and integrity costs with external data sources, instead of using agents to import RDF data to semantic network, web services are used.
In this manner, the new SWT based system framework substitutes some data sources with web services that provide data on demand. This way, dynamic data as defined earlier, is being provided by web services. To be able to combine static and dynamic data in a transparent manner, the framework has to be able to automatically execute Web Services, when the need for data they provide arises. To solve the above problem we propose to incorporate SOA (Service oriented architecture) principles (Erl, 2005).

4. Incorporating SOA concepts

Though SWT are very suitable for integration purposes, we identified some short-comings in this approach. SWT are very flexible and we can transform practically any machine readable data into RDF and then process it based on the ontology. To be able to integrate data, we have to import it into the semantic network first. When the intention is not to reason on integrated data, but rather just to integrate it, there are other more suitable methods for data integration. Because of the fact, that we have to import the data into the semantic network to be able to integrate it, there arise two main problems: (1) the size of the semantic network can grow at a very fast rate, because of this there can be performance issues and (2) by importing data into the semantic network, we basically replicate the data. In order to have accurate information, we have to synchronize the data between the originating data source and the semantic network.
In a SWT system, we can have information, that is crucial for the reasoning process and also information that is not used for the reasoning process. This kind of information is used for

providing additional information to the user. For example, suppose we want to support medical decision making process of selecting the most suitable physician for a support request. To be able to reason about the physician's competences, we need information about his previous experiences and education. To be able to propose a suitable physician, we need information about the nature of the problem and competences of all the physicians.
Let us suppose that this information is sufficient for successful physician selection. To be able to propose a physician, we have to import this information into semantic network. While this information is sufficient for the reasoning process, the user, that uses knowledge management system may have needs for additional information like contact information of the patient requesting physician, previous records for this particular patient etc. Usually this information is already stored in some other information system and synchronization can result in high development costs and larger amount of processing.
Therefore we propose use of services to provide information that is not essential to reasoning process. Information space can be divided into two subsets: static and dynamic data. Static data is data that is needed for the reasoning process and should be always available in the semantic network. Dynamic data, on the other hand, is not needed by the reasoning process – it can be provided dynamically when the need for it arises. In this manner, we propose the use of services to provide these dynamic data (Figure 3). That way we don't need to import a large amount of data into the semantic network. Instead, the data is being requested when the need for it arises. If we return to our example, when the knowledge system proposes a particular physician, if the user has need to contact the patient (physician requester), the service that provides patient contact information can be automatically executed and the user can be seamlessly presented with contact information.
In a SWT system the data is described semantically. This means that computers have an awareness of the meaning of data. In the same manner we can semantically describe services. If we model and integrate semantics of the data and services in an ontology, the services can be executed automatically. That way information can be presented to the user transparently in both cases, whether it is stored in the semantic network or provided by a service. Relations between data semantics and its role in organization knowledge can be modelled in an ontology.

4.1 The principles and benefits of SOA

Service oriented architecture (SOA) is an evolutionary step from enterprise application integration (EAI). Main purpose and goal of both EAI and SOA is integration of information between several different information systems. EAI creates tight connections between different information systems in such way, that each information system imports data from other information systems. While this is fairly natural way of integration of information, it is often connected with high maintenance efforts and low level of reuse and flexibility.
Opposite to tightly coupled EAI, SOA introduces loosely coupled, distributed and flexible architecture. SOA introduces concepts of services. While EAI creates connections between each pair of applications, applications in SOA environment expose their data and also functionality in form of platform and program language independent services. Applications that need information or other applications functionality only have to query the service. By also exposing functionality greater level of reuse can be achieved.
According to (Erl, 2005), the key aspects of SOA are: (1) loose coupling – services maintain a relationship that minimizes dependencies and only requires that they retain an awareness of

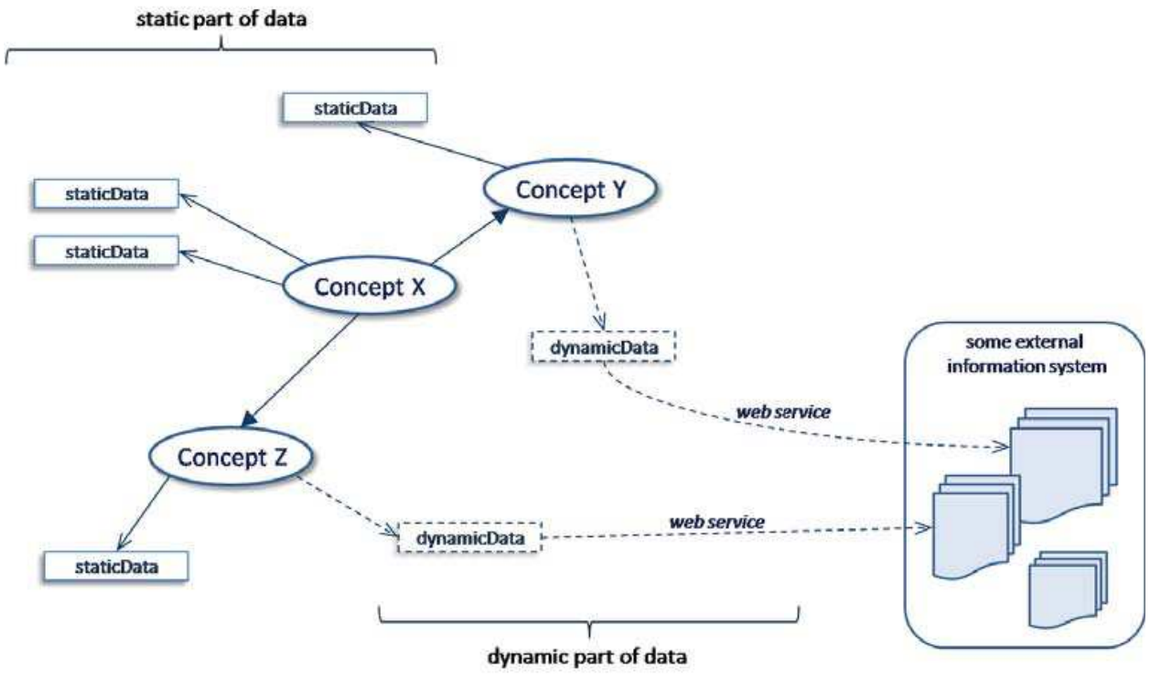


Fig. 3. Basic concept of decoupling semantic information into static and dynamic (provided by an external information system) data

each other; (2) service contract – a communications agreement, as defined collectively by one or more service descriptions and related documents; (3) autonomy – services have control over the logic they encapsulate; (4) abstraction – beyond what is described in the service contract, services hide logic from the outside world; (5) reusability – logic is divided into services with the intention of promoting reuse; (6) composability – collections of services can be coordinated and assembled to form composite services; (7) statelessness – services minimize retaining information specific to an activity, (8) discoverability – services are designed to be outwardly descriptive so that they can be found and assessed via available discovery mechanisms.

SOA is basically a concept and is independent of the implementation technologies. When the term service oriented architecture is used, it is mostly meant in a context of web services. In this manner, (Erl, 2005) defines contemporary SOA as SOA, that is built around web service technologies. When we use the term SOA in this paper, we actually mean contemporary SOA as defined by (Erl, 2005).

The architecture, presented in the following sections, can be seen as an extension of SOA. The same service can be reused for automated ontology based data integration, as described in this paper, as well as in other common SOA aware applications.

4.2 Semantic web services execution ontology (SWSEO)

Web services (WS) technology mainly provides standards for functional and also nonfunctional description, e.g. quality of service, security, authorization. While they provide very good technical platform, they lack in providing semantics to the services. This results in worse than expected discovery, reuse and composition of web services. To overcome these issues, semantic web services (Akkiraju, 2006) were introduced.

Semantic web services (SWS) combine concepts from semantic web and web services. As Grosof (2003) pointed out, semantic web services can be understood in two ways: as (1) semantic [web services] and as (2) [semantic web] services. The former concept relies more on WS and is used for knowledge-based service descriptions (discovery, execution, composition of WS), while the latter concentrates more on SW concepts and is used mainly for knowledge and information integration. In our framework we use SWS in both contexts, as S[WS] for capturing IT support knowledge and as [SW]S for supporting representation of operational knowledge (classic KMS).

There are several approaches for implementing SWS, the most common are: WSMO, OWL-S and SAWSDL. There are slight differences in basic concepts. We chose to use SAWSDL because of the following facts: (1) we don't need complex discovery and composition capabilities in our framework; thus we can use a simpler formalism, (2) WSMO uses its own language that is not compatible with SWT, (3) research and development effort of OWL-S is fading and a lot of tools are already outdated, (4) existing WS can be easily converted to SAWSDL by just semantically annotating WSDL, (5) SAWSDL is not just compatible with SWT but it is also compatible with current Web Services, (6) SAWSDL is interoperable with SOA implementations.

SAWSDL does not specify how services are modeled. Because of that, we have developed a lightweight service modeling ontology that is targeted at automated execution of web services. The ontology is called semantic web services execution ontology (SWSEO). It is defined in OWL-DL and is shown in Figure 4.

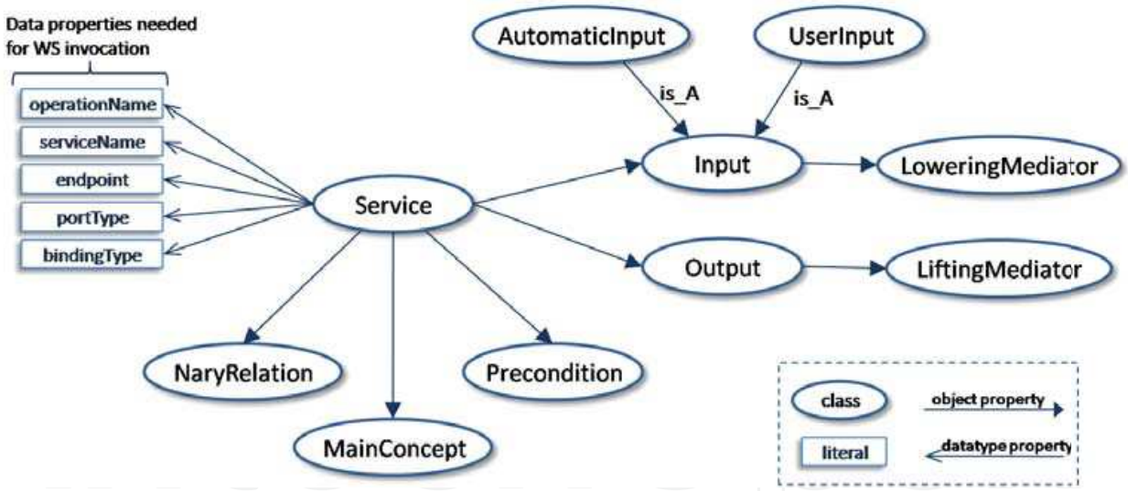


Fig. 4. Basic concepts of Semantic Web Services Execution Ontology (SWSEO)

Main concept in SWSEO is service. Opposite to WSDL, SWSEO service is actually a WSDL operation. It has five data properties (presented on the left hand side in Figure 4) which are used for web service invocation and are parsed from SAWSDL document at the registration time.

Every service has also some input and output messages. Web Services are based on XML language and use XML documents as input and output messages to services. To be able to use this data in semantic networks; XML data has to be converted to RDF format and vice versa. In case of input messages, semantic data represented in RDF has to be lowered to XML documents and in case of output messages XML data has to be lifted to RDF format.

For this purpose we defined lifting and lowering mediators that take care of data conversion.
Lifting mediators can be XSL transformations (Kay, 2007), Java classes or even other Web Services. Lowering mediators on the other hand have one additional mediator type: SparQL mediator. In RDF the same data can be represented in different ways. Because of that, XSLT, which does transformation based on XML document structure, cannot handle all different data representation variations. For this purpose a combination of SparQL query and XSLT can be used. SparQL takes care of getting the right structure, after that XSLT is used to actually lower the data. As seen in Figure 3, input messages can be of two types: (1) automatic input – data is being fetched automatically from the semantic network by the framework and (2) user input – input data is not stored in the semantic network, but it is provided by the user when querying semantic network.
There are three further concepts, we haven't mentioned yet. First is precondition. This concept is used in case more than one service provides the same type of data and the service that is being invoked is selected based on the instance, e.g. let us suppose we have an e-health application that uses web services for getting some medical equipment availability information.
Each participating equipment provider provides its own availability service; all the availability services provide same type of data (equipment availability). Service that is being invoked is selected based on precondition. Precondition defines for which participating provider a particular service provides medical equipment availability information.
Concept named main concept is used for extracting automatic input messages from semantic network and for creating service execution plan. For example described in previous paragraph, main concept would be participating medical equipment provider. This means that the framework has to check equipment availability by executing the web service for each given provider.
Last basic concept of SWSEO is nary relation. RDF and OWL support only binary relations between concepts. If we want to make a nary relation, we have to create a holding class (Hayes & Welty, 2006). There are situations where services provide nary relations for which holder class instances are not yet in the semantic network. This class actually serves as an instruction for the framework to create the holding class instance and connect it with the main concept.

4.3 Automated semantic web services execution architecture

Figure 5 shows system architecture for automated execution of SWS in accordance with SWSEO. The system acts like a wrapper to the SparQL endpoint. The whole process of service input retrieval, data conversion, service execution and model integration is transparent to the user; the user has to provide only the SparQL query.
Main components of the architecture are: (1) SparQL query processor – it splits a query into static and dynamic part, returns concepts from query that are service outputs and parses user service input information from where clauses, (2) input provider – provides data defined as automatic input, (3) input and output mediators – provide lifting and lowering capabilities (Java, XSLT, and SparQL mediators), (4) service executor – invokes web services, (5) query executor wrapper – provides data from semantic networks needed by other components in an efficient way by caching data and reducing the number of query executions.

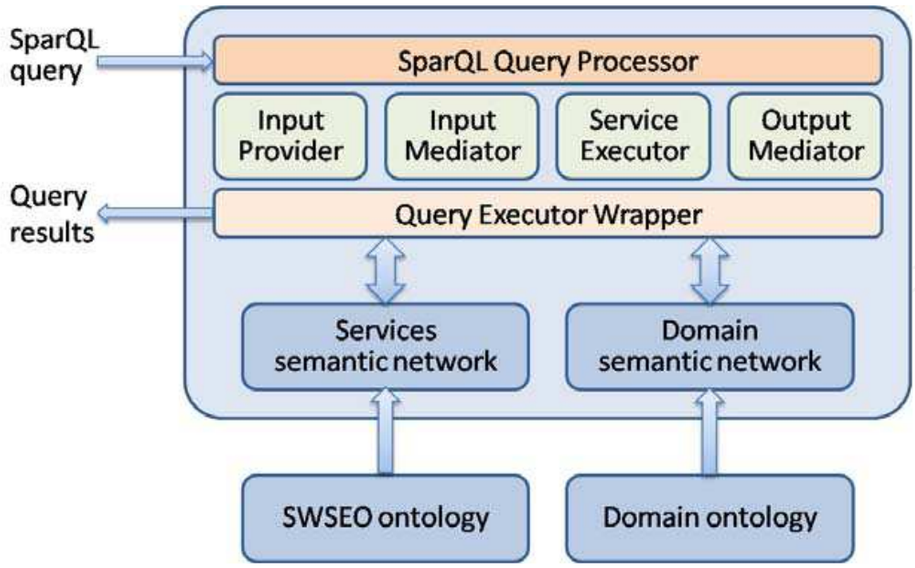


Fig. 5. Automated Semantic Web Services execution architecture

5. Improved SWT based system architecture

Based on all the concepts discussed above, we can now finally represent the improved system architecture framework based on SWT that should be a possible solution to the idea of the unified information system architecture for software systems proposed in the introduction of this chapter. The conceptual system architecture is presented on Figure 6.

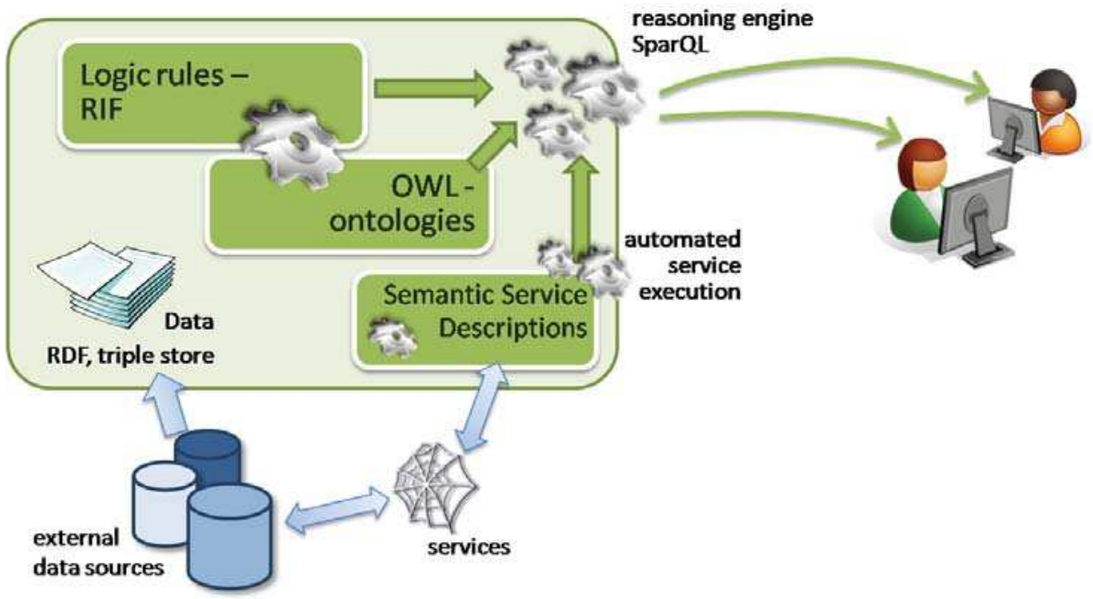


Fig. 6. Advanced SWT based system architecture

The presented system can collect data from both static data repositories (relational databases, intranet site, file systems, etc) and also from various web services. In this manner, the integration with other information systems should be easier to achieve, especially when

their data have already been exposed in a form of web services. By exposing data as web services also the size of the semantic network can be reduced, which results in better performance. In order to be able to fully use the advantages of semantic based system, the services should be described with a proper semantic service description technology (as discussed previously).
The proposed architecture is driven by a set of specific ontologies for different views within the whole medical process, which should be interconnected in order to fully explore the possibilities of semantic data. The system is open to connect to various publicly available semantic databases, if it is appropriate for a user. The processing of the data in the system is by no means limited to a single technology. All known technologies, tools and methods for efficient processing of semantic data can be used.
The reasoning engine that provides the data to users over different user interfaces (web or desktop applications, data repositories endpoints . . .) is defined on a set of standardized technologies, such as logic rules (RIF), SparQL query language, etc. In this manner, the final processing system can be defined in such a way that best suits the software requirements of a specific implementation.

6. A case study: project team building

To test the appropriateness of the presented improved SWT based system architecture for implementing an innovative IT solution we decided to develop a system for supporting the building of project teams regarding the requirements of a project and skills of potential team workers. For a technical project, let's say in software engineering, to be successfully implemented there is a need for bright, skilled individuals with good technical skills and exceptional attention to detail. However, the real world projects nowadays normally require more than just good individuals. Even a group of great individuals is not enough. What we really need is a team – a team of cleverly selected individuals, who will combine their personal technical skills with their teamwork skills in order to achieve the project goals. What we need to compose great teams is a proper team building approach and a supporting technology to implement the approach. It is our belief that the improved SWT based system architecture, presented in previous sections, is a good and valid approach to project teams building.

6.1 Team building overview

Team building is an effort in which a team studies its own process of working together and acts to create a climate that encourages and values the contributions of team members. Their energies are directed toward problem solving, task effectiveness, and maximizing the use of all members' resources to achieve the team's purpose. Sound team building recognizes that it is not possible to fully separate one's performance from those of others. Team building works best when the following conditions are met (Frances & Young, 1979):

- There is a high level of interdependence among team members. The team is working on important tasks in which each team member has a commitment and teamwork is critical for achieving the desired results.
- The team leader has good people skills, is committed to developing a team approach, and allocates time to team-building activities. Team management is seen as a shared function, and team members are given the opportunity to exercise leadership when their experiences and skills are appropriate to the needs of the team.

- Each team member is capable and willing to contribute information, skills, and experiences that provide an appropriate mix for achieving the team's purpose.
- The team develops a climate in which people feel relaxed and are able to be direct and open in their communications.
- Team members develop a mutual trust for each other and believe that other team members have skills and capabilities to contribute to the team.
- Both the team and individual members are prepared to take risks and are allowed to develop their abilities and skills.
- The team is clear about its important goals and establishes performance targets that cause stretching but are achievable.
- Team member roles are defined, and effective ways to solve problems and communicate are developed and supported by all team members.
- Team members know how to examine team and individual errors and weaknesses without making personal attacks, which enables the group to learn from its experiences.
- Team efforts are devoted to the achievement of results, and team performance is frequently evaluated to see where improvements can be made.
- The team has the capacity to create new ideas through group interaction and the influence of outside people. Good ideas are followed up, and people are rewarded for innovative risk taking.
- Each member of the team knows that he or she can influence the team agenda. There is a feeling of trust and equal influence among team members that facilitates open and honest communication.

Team building will occur more easily when all team members work jointly on a task of mutual importance. This allows each member to provide their technical knowledge and skills in helping to solve the problem, complete the project, and develop new programs. During this process, team building can be facilitated as members evaluate their working relationship as a team and then develop and articulate guidelines that will lead to increased productivity and team member cooperation. Team performance can best be evaluated if the team develops a model of excellence against which to measure its performance.

6.2 SWT based team building approach

We decided to develop the whole system in a manner of a semantic web portal, which serves as an entry point to our solution in project team building. It supports users in building project teams effectively and efficiently. The architecture of the portal is presented in Figure 7.

This architecture provides a mean to manage both members' and projects' profiles through a web server by members themselves, by project leaders and by project administrators. The inference engine uses the profiles together with previous projects' data in order to propose members for any new project regarding the requirements. The system's inferring capabilities can be improved by managing the skills matching database which is used to reveal the hidden skills of members, not provided directly by them or the project leaders.

During the project cycle the portal should not be used too frequently. If we would push users to use it over and over again, we would disturb project activities and waste team's energy. However, we want results from the portal – so we designed it to be used only on beginning and ending of a project (Figure 8). If project team uses information support for their project activities, portal should integrate their existing data, so no changes in work are necessary. When using some groupware to support project work, the data could be gathered automatically.

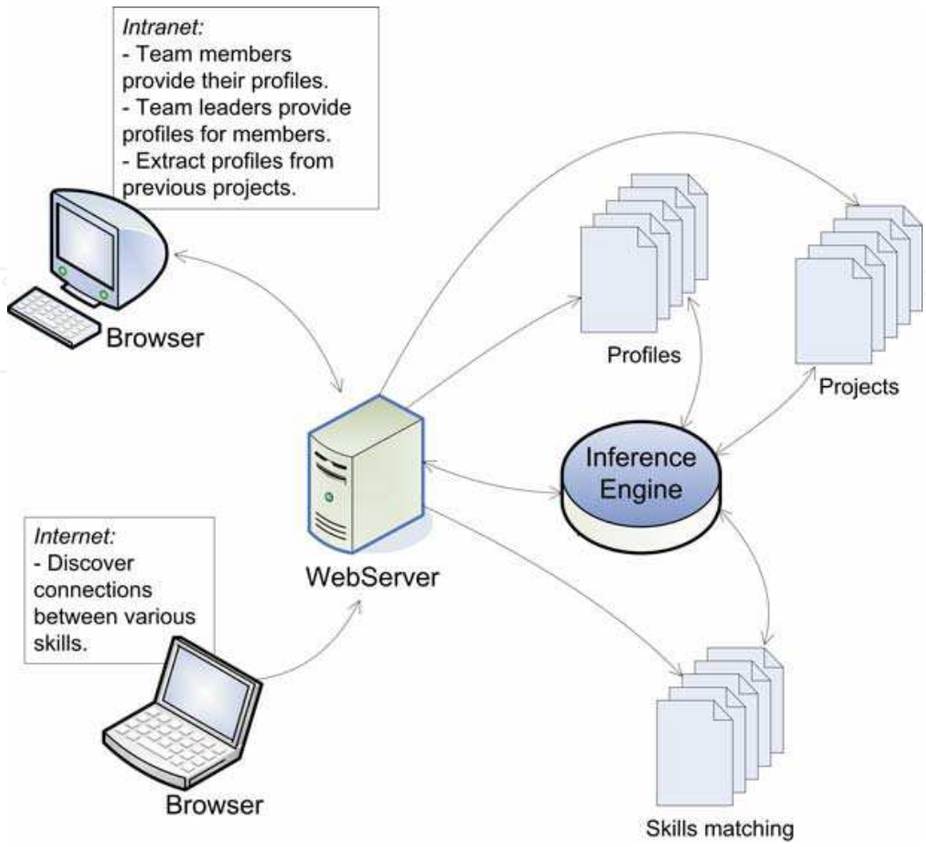


Fig. 7. The architecture of a semantic web portal for project team building

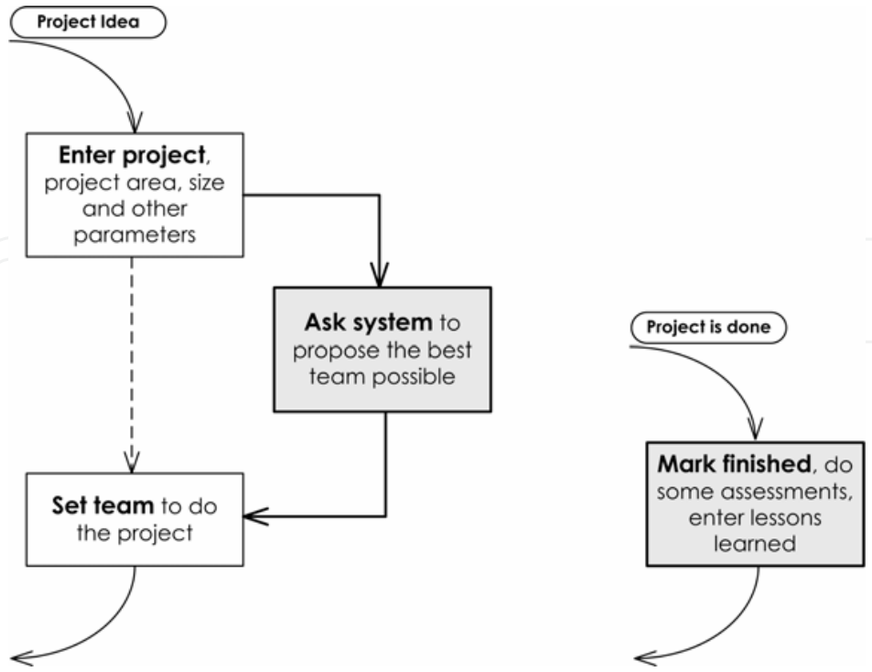


Fig. 8. The role of portal for project team building during project cycle

6.3 Ontology-based personal skill management

An overview of the related work in ontology-based personal skill management is presented in Biesalski & Abecker (2005). Already Stader & Macintosh (1999) and Jarvis et al. (1999) promoted the idea of ontology-based modelling of personnel skills and job requirements – as part of comprehensive, workflow-oriented enterprise modelling. There, the following potential applications of ontology-based skill profiles are listed:

- skill gap analysis – at the enterprise level, as a part of strategic HR planning,
- project team building,
- recruitment planning – again a part of strategic HR planning,
- training analysis – at the level of individual personnel development.

Those approaches were mainly technology-driven and were – to our knowledge – never realized in a large-scale industrial environment. Nor have they been accepted by the HRM departments, translated into HRM people's terminology, embedded into more comprehensive models and procedures of HRM people, and integrated with existing software infrastructures. After those first publications, there were a number of interesting technology-oriented researches which showed that in particular skill matching can benefit from interesting technological approaches, such as background knowledge exploitation. For instance, Liao et al. (1999) employs declarative retrieval heuristics for traversing ontology structures. Sure et al. (2000) derives competency statements through F-Logic reasoning and developed a soft matching approach for skill profile matching. Colucci (Colucci et al. (2003)) use description logic inferences to take into account background knowledge as well as incomplete knowledge when matching profiles.

6.4 Application ontology: personal skills and project team

There have been many approaches to describe personal skills within an ontology. They included both technical skills (like knowledge of programming languages, development methods, specific tools) and inter-personal skills (like communication skills, affableness, teamwork). Based on the team building theory and our own experiences from performed projects the main items that have to be included in such ontology should be:

- formal and informal education,
- experiences,
- practical skills,
- performed projects,
- preferred tasks,
- preferred role within a team,
- communication skills,
- teamwork spirit.

As far as we could find within the recent literature, there have been some approaches to describe project teams with an ontology. However, they have not been used for project teams building. For this purpose the existing project team ontologies, which include basic information about team members, resources, purpose of the project, etc., should be extended with the following information:

- the priorities of the project,
- the importance of specific phases (regarding the current stage),
- complexity of the project (regarding the previous ones),
- the technical type of the project (prototype, research, production, etc),
- special knowledge requirements,

- preferred personnel,
- the type of the product/service being developed,
- the relation with other (especially previously successfully performed) projects.

The resulting ontology is presented in Figure 9 – only classes and object properties are shown. Ontology is used in the portal to help reasoner use metadata and construct the proper team for performing the project.

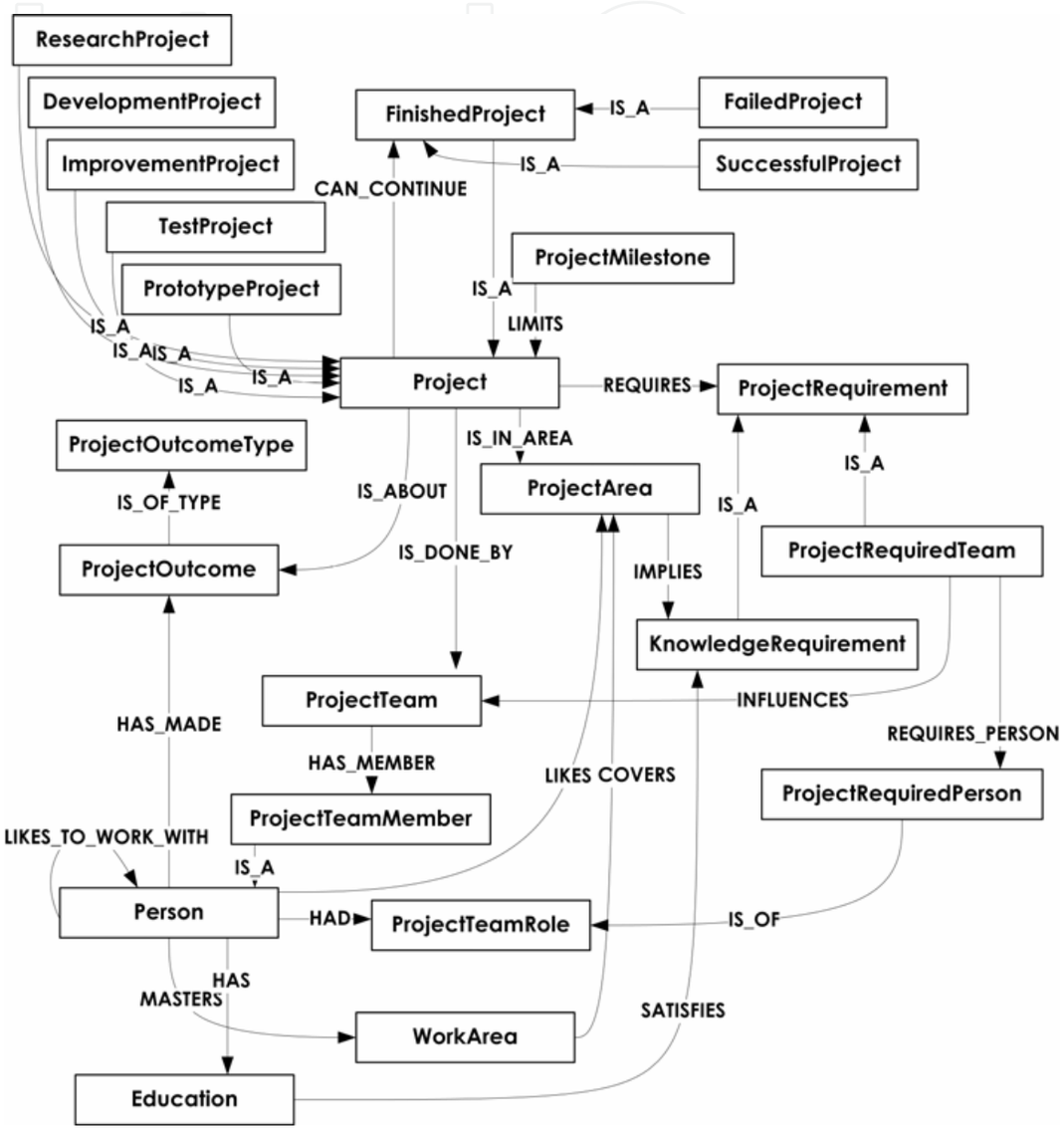


Fig. 9. The used ontology for project team members selection

6.5 Inferring on the application ontology

The project team building system prototype is implemented mainly in Java programming language using open source Jena semantic web development library[3]. It provides us with a

[3] http://jena.sourceforge.net

straight-forward development system, very appropriate for semantic web enabled applications. For the inferring part, we have used Pellet[4]. Pellet is an OWL reasoner for Java that enables monotonic reasoning on rules specified in semantic web rule language (SWRL). Using rules on top of a formal ontology enables powerful inferring capabilities.
To demonstrate rule based inference, example pair of rules are shown in Figure 10. These two rules select suitable persons for a project team based on project requirements. Concepts that are used in the rules are defined in the ontology above (Figure 9). In the example we can see that project team members can be selected based on the project requirements in two ways:

- using the information about education of a person (knowledge requirements that are covered by some education), and
- using the information about work areas mastered by a person (a work area covers several project areas and a project area implies several knowledge requirements).

$$IS\_DONE\_BY(?project, ?projectTeam) \wedge REQUIRES(?project, ?knowledgeReq) \wedge HAS(?person, ?education) \wedge SATISFIES(?education, ?knowledgeReq) \rightarrow HAS\_MEMBER(?projectTeam, ?person)$$

$$IS\_DONE\_BY(?project, ?projectTeam) \wedge REQUIRES(?project, ?knowledgeReq) \wedge MASTERS(?person, ?workArea) \wedge COVERS(?workArea, ?projectAres) \wedge SATISFIES(?projectArea, ?knowledgeReq) \rightarrow HAS\_MEMBER(?projectTeam, ?person)$$

Fig. 10. An example set of SWRL rules for inferring on the ontology.

In this way a team member can be automatically selected based on the requirements of the project. Note that an IS_A relation between knowledge requirement and project requirement has been implicitly used in this example. The knowledge base for inferring on SWRL rules is induced directly from the ontology and/or the corresponding database.
In order to create a holistic application for managing projects and team members, the application has to provide complete information about project members and projects as well. Unfortunately, this kind of information is usually stored in legacy applications. It is not reasonable to replicate all this data in semantic application. For this reason, the prototype uses services to obtain information from other applications in the manner of service oriented architecture.
Semantics of the data that is returned by Web Services is modelled in the application ontology. SWSEO based service annotations enable reasoning system to obtain data dynamically by automatically executing Web Services. Besides proposing appropriate team members as described earlier, SWSEO compatible services on the other hand enable us to dynamically integrate detail information about project members from other systems (e.g. personal information, detailed project description as well as work items, human resources availability, team member's assignments, passed certifications, etc.)

6.6 Technologies used

As shown in Figure 11 the system itself does not consist of many different technologies, which is in our belief good. As mentioned before, fundamentals for the system lies in J2EE

[4] http://clarkparsia.com/pellet

platform, XML enabled database (we used the Oracle 10gR2 database), and connection with the external web services.

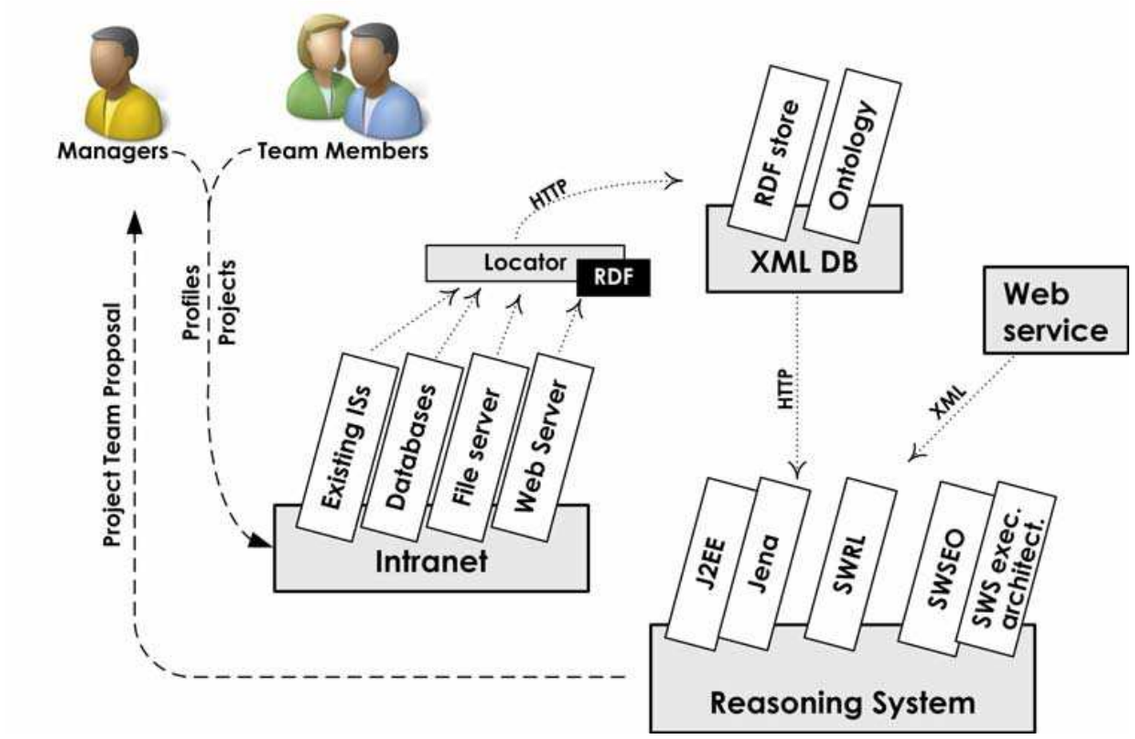


Fig. 11. The main technologies used to develop the project team building portal

First vital component, called "locator", is responsible for collecting as many internal data in RDF as possible. It extracts data from existing systems, web pages, databases and file systems. Collected RDFs and presented ontology are persistently stored in XML database and represent the static part of the whole semantic data network. The dynamic part of the data is served on demand by the web service. The both parts together are prepared for the reasoning system, based on J2EE and Jena with the use of SWRL. So the second vital component is the reasoning system, which uses the SWSEO-enabled web services, automated semantic web services execution architecture and SWRL rules to infer on the integrated semantic data.

7. Conclusions

This paper provides some results of our endeavour in adopting SWT for implementing innovative IT solutions. Performing some experiments using SWT as an enabling technology, we came to the conclusion that the common SWT based system architecture has some important drawbacks. In this manner, knowing the great potential of SWT, our goal was to find out whether improved settings of SWT can be able to overcome at least some of the difficulties encountered.

The proposed approach to the utilization of state of the art software technologies for the development of innovative IT solutions using SWT serves the purpose. In some pilot implementations the proposed improved system architecture enabled us to integrate the data from different processes at the ontology level. Furthermore, it enabled us to

interconnect our own ontology with the existing ontologies, with both semantic and relational data repositories, and also with the dynamic data from web services. Using the division of semantic data between static RDF based data repositories and external semantic web services, we succeeded to somewhat reduce the poor performance of known semantic applications. It must be said, however, that only a very limited data resources have been used in our experiments, which show no exact proof of how the system would perform when scaled to a real world software system. By exposing data as web services, the size of semantic network can be reduced, which results in better performance. Yet this does not have a deterministic behaviour and further research work should be done on this topic. On the other hand by enabling integration and automatic execution of web services better interoperability with other information systems can be achieved.

It is our intention to further improve the proposed SWT based system architecture in the future. Primarily, we would like to perform some tests on scaling properties of the proposed system architecture. Furthermore, the integration possibilities with the existing information systems, both monolithic and service based, should be evaluated in a more detailed manner. Finally, we would like to test the system in a real world environment using a defined methodology in order to evaluate whether the technology is appropriate to be used by professional software developers.

8. References

Akkiraju, R. (2006). *Semantic Web Service - Theory, Tools, and Applications*, Idea Group, chapter Semantic Web Services, pp. 191–216.

Berners-Lee, T., Hendler, J. & Lassila, O. (2001). The semantic web: Scientific american, *Scientific American* .
URL: http://www.scientificamerican.com/article.cfm?id=the-semantic-web

Biesalski, E. & Abecker, A. (2005). Integrated processes and tools for personnel development, *1th International Conference on Concurrent Enterprising, University BW Munich, Germany, 20-22 June 2005.*

Colucci, S., Noia, T. D., Sciascio, E. D., Donini, F. M., Mongiello, M. & Mottola, M. (2003). A formal approach to ontology-based semantic match of skills descriptions, *Journal of universal computer science, Special issue on skills management* 9: 1437–1454.

Dean, M. & Schreiber, G. (2004). OWL web ontology language reference, *W3C recommendation*, W3C.

Erl, T. (2005). *SOA Principles of Service Design*, Prentince Hall/PearsonPTR.

Frances, D. & Young, D. (1979). *Improving Group Work: A Practical Manual for Team Building*, University Associates, Inc.

Grosof, B. (2003). Semantic web services: Obstacles and attractions, *Panel at 12th Intl. Conference on WWW.*

Hayes, P. & Welty, C. (2006). Defining n-ary relations on the semantic web, W3C Working Group Note.

Jarvis, P., Stader, J., Macintosh, A., Moore, J. P. & Chung, P. W. H. (1999). What right do you have to do that?-infusing adaptive workflow technology with knowledge about the organisational and autority context of a task., *ICEIS*, pp. 240–247.

Kay, M. (2007). Xsl transformations (xslt) version 2.0 (w3c recommendation 23 january 2007), *Technical report*, W3C.
URL: http://www.w3.org/TR/xslt20/

Knaup, P., Garde, S. & Haux, R. (2007). Systematic planning of patient records for cooperative care and multicenter research, *International Journal of Medical Informatics* 76(2-3): 109 – 117. Connecting Medical Informatics and Bio-Informatics - MIE 2005.

Lenz, R., Beyer, M. & Kuhn, K. A. (2007). Semantic integration in healthcare networks, *International Journal of Medical Informatics* 76(2-3): 201 – 207. Connecting Medical Informatics and Bio-Informatics - MIE 2005.

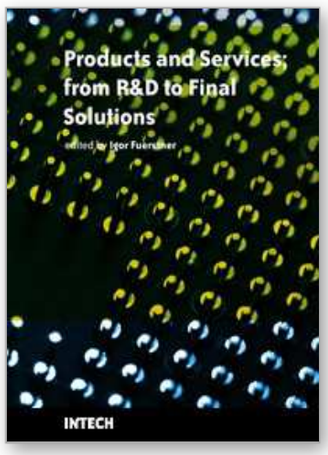
Liao, M., Hinkelmann, K., Abecker, A. & Sintek, M. (1999). A competence knowledge base system as part of the organizational memory, *XPS '99: Proceedings of the 5th Biannual German Conference on Knowledge-Based Systems*, Springer-Verlag, London, UK, pp. 125–137.

Manola, F. & Miller, E. (eds) (2004). *RDF Primer*, W3C Recommendation, World Wide Web Consortium.
URL: http://www.w3.org/TR/rdf-primer/

Passin, T. B. (2004). *Explorer's Guide to the Semantic Web*, Manning Publications Co.

Stader, J. & Macintosh, A. (1999). Capability modelling and knowledge management, *Applications and Innovations in Expert Systems VII, Proceedings of ES 99 the 19th International Conference of the BCS Specialist Group on Knowledge-Based Systems and Applied Artificial Intelligence*, Springer-Verlag, Berlin, pp. 33–50.

Sure, Y., Maedche, A. & Staab, S. (2000). Leveraging corporate skill knowledge - from proper to ontoproper, *in* D. Mahling & U. Reimer (eds), *Proceedings of the Third International Conference on Practical Aspects of Knowledge Management. Basel, Switzerland, October 30-31, 2000*. http://www.research.swisslife.ch/pakm2000/.



Products and Services; from R&D to Final Solutions
Edited by Igor Fuerstner

ISBN 978-953-307-211-1
Hard cover, 422 pages
Publisher Sciyo
Published online 02, November, 2010
Published in print edition November, 2010

Todayâ€™s global economy offers more opportunities, but is also more complex and competitive than ever before. This fact leads to a wide range of research activity in different fields of interest, especially in the so-called high-tech sectors. This book is a result of widespread research and development activity from many researchers worldwide, covering the aspects of development activities in general, as well as various aspects of the practical application of knowledge.

How to reference
In order to correctly reference this scholarly work, feel free to copy and paste the following:

Vili Podgorelec and Bostjan Grasic (2010). Implementing Innovative IT Solutions with Semantic Web Technologies, Products and Services; from R&D to Final Solutions, Igor Fuerstner (Ed.), ISBN: 978-953-307-211-1, InTech, Available from: http://www.intechopen.com/books/products-and-services--from-r-d-to-final-solutions/implementing-innovative-it-solutions-with-semantic-web-technologies

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen