Chapter II

A Semantic Service-Oriented Architecture for Business Process Fusion

Athanasios Bouras, National Technical University of Athens, Greece
Panagiotis Gouvas, National Technical University of Athens, Greece
Gregoris Mentzas, National Technical University of Athens, Greece

Abstract

Most enterprises contain several heterogeneous systems, creating a fuzzy network of interconnected applications, services, and data sources. In this emerging business context, a clear need appears to link these former incompatible systems by using enterprise application integration (EAI) solutions. We propose a semantically enriched service-oriented business applications (SE-SOBA) framework that will provide a dynamically reconfigurable architecture enabling enterprises to respond quickly and flexibly to market changes. We also propose the development of a pure semantic-based implementation of the universal description, discovery, and integration (UDDI) specification, called pure semantic registry (PSR), which provides
a flexible, extendable core architectural component allowing the deployment and business exploitation of Semantic Web services. The implementation of PSR involves the development of a semantic-based repository and an embedded resource definition framework (RDF)-based reasoning engine, providing strong query and inference capabilities to support effective service discovery and composition. We claim that when SE-SOBAs are combined with PSR and rule-based formalizations of business scenarios and processes, they constitute a holistic business-driven semantic integration framework, called FUSION, applied to intra- and inter-organizational EAI scenarios.

Introduction

In today’s fiercely competitive global economy, companies are realizing that new initiatives such as e-business, customer relationship management, and business intelligence go hand-in-hand with the proven organization-wide EAI strategy. The goal of EAI is to integrate and streamline heterogeneous business processes across different applications and business units while allowing employees, decision makers, and business partners to readily access corporate and customer data no matter where it resides. More and more, EAI involves integrating information and processes not only across the enterprise but also beyond organizational walls to encompass business-to-business (B2B) integration supporting large scale value-added supply chains across the enlarged worldwide economy.

Business process fusion is the transformation of business activities that is achieved by integrating the interfaces of previously autonomous business processes by pipelining different middleware technologies and enabling the effective (semi-)automated exchange of information between various systems within a company or between enterprises. The development of SOBAs (which constitutes a set of independently running services communicating with each other in a loosely coupled message-based manner) and the publishing of Web services may implement the vision of business process fusion, by providing an abstraction layer for the involved interfaces through the Web service description language (WSDL). While SOBA and Web services have already made headway within large organizations, the technology will start filtering down to small- and medium-sized enterprises (SMEs) and will expand into supply chains. This architecture will also play a significant role in streamlining mergers and acquisitions, by linking previously incompatible systems.

Despite the aforementioned trends, users and professionals have high expectations towards software applications and enterprise application integration. They want to access the content they need, while this content must be accurate and free of redundancy. So, the enterprise applications must be intuitive and easy to use; reus-
able and extendable; implemented in a short and inexpensive way; and within the current information technology (IT) legacy environment. Enterprise applications and information systems also need to support a more general notion that involves relating the content and representation of information resources to entities and concepts in the real world.

This need imposes the use and interpretation of semantics in EAI. Semantic interoperability will support high-level, context-sensitive, information requests over heterogeneous information resources, heterogeneous enterprise applications, hiding systems, syntax, and structural heterogeneity. This semantically enriched approach eliminates the problem of knowing the contents and structure of information resources and the structure and architecture of heterogeneous enterprise applications.

Semantics and ontologies are important to application integration solutions because they provide a shared and common understanding of data, services, and processes that exist within an application integration problem domain, and how to facilitate communication between people and information systems. By leveraging this concept we can organize and share enterprise information, as well as manage content and knowledge, which allows better interoperability and integration of inter- and intra-enterprise information systems.

We claim that recent innovations in the development of SE-SOBA—which enlarge the notion of service-oriented architecture (SOA) by applying Semantic Web service technology and using ontologies and Semantic Web markup languages to describe data structures and messages passed through Web service interfaces—combined with the rule-based formalization of business scenarios and processes will provide a dynamically reconfigurable architecture that will enable enterprises to respond quickly and flexibly to market changes, thereby supporting innovation and business growth, increasing the potential for an improved return on IT investments, and a more robust bottom line.

The structure of this chapter is as follows: in the following section, we define the concept of EAI and present the traditional and current trends of EAI from the technology perspective. In the section called The Road to Enterprise Application Integration, we present the way that the emerging Semantic Web technologies apply to EAI scenarios and analyze the state-of-the-art technologies and techniques. The conceptual framework, called FUSION, which we propose referring to the innovative business-driven, semantic-enriched, service-oriented architecture, as well as the proposed business-oriented ontologies that extends OWL-S Service Profile are defined in the next section, called FUSION Conceptual Framework, while the technical implementation of our approach is presented in FUSION Technical Implementation. Moreover, the section FUSION Adoption: Integration Scenario and Applying Methodology specifies a light FUSION adoption methodology and a typical application scenario of the proposed solution. Finally, we present further work; future trends and technologies; and concluding remarks.
The Road to Enterprise Application Integration

Traditional Enterprise Application Integration

Most enterprises contain a systemic infrastructure of several heterogeneous systems, creating a complex, fuzzy network of interconnected applications, services, and data sources, which is not well documented and expensive to maintain (Samtani & Sadhwani, 2001). Moreover, the introduction of multi-oriented, separate legacy systems concerning enterprise resource planning (ERP), customer relationship management (CRM), supply chain management (SCM), e-business portals and B2B transactions, increases the complexity of systems integration, making the support of the interoperability among these systems a challenging task.

In this emerging business context, a clear need appears to link these former incompatible systems to improve productivity and efficiency. The solution to this need is what is called EAI, which can be defined as the use of software and architectural principles to bring together (integrate) a set of enterprise computer applications (see Figure 1). The goal of EAI is to integrate and streamline heterogeneous business processes across different applications and business units. We distinguish between intra- and inter-organizational enterprise application integration. Intra-organizational EAI, commonly referred as Application to Application Integration (A2A) (Bussler, 2003a), specifies the automated and event-driven exchange of information between heterogeneous enterprise applications and systems operating within an organization or enterprise. On the other hand, inter-organizational EAI, or else B2B integration (Bussler, 2003a), specifies the automated and event-driven information exchange between various systems of several collaborating organizations and enterprises.

Figure 1. The enterprise system environment: With and without an EAI system
Moreover, Apshankar et al. (2002) identify different types of EAI levels/layers, explaining the various dimensions of the integration task, namely:

- data-oriented integration, occurring at the database and data source level, either real time or non-real time, constituting the most widespread form of EAI today;
- function or method integration, involving the direct and rigid application-to-application integration of cross-platform applications over a network—it can be achieved using custom code, application program interface (APIs), remote procedure calls (RPCs) or distributed middleware and distributed objects (CORBA, RMI, DCOM);
- user interface integration, consisting on using a standardized user interface for accessing a group of legacy systems and applications. The new presentation layer is integrated with the existing business logic of the legacy systems or packaged applications; and
- business process integration, occurring at the business process level.

In recent years, most enterprises and organizations have made extensive investments in several EAI systems and solutions that promise to solve the major integration problem among their existing systems and resources. The business driver behind all these traditional EAI projects is to integrate processes across third-party applications as well as legacy systems to decrease the number of adapters one has to develop if connecting two systems (Laroia & Sayavedra, 2003). Therefore, the traditional EAI focuses (Haller, Gomez, & Bussler, 2005) on the message-based communication of software applications interfaces, by pipelining different middleware technologies and developing various adapters, connectors, and plug-ins to provide efficient messaging support among heterogeneous systems, allowing their effective interconnection. However, traditional EAI efforts lack of an upper abstraction layer, as well as standardized architectures and implementations, making customers and end users captive of EAI vendor-specific solutions, and arising a new, high-level integration problem of interconnecting various EAI systems with one another. The growth of the EAI market and the involvement of new EAI vendors have intensified the integration problems identified, considering the standardization of integration frameworks and architectures a necessity. The development and introduction of Web service enabled service-oriented architecture solutions, completely based on widely known and accepted standards, overcomes the aforementioned EAI obstacles.
Web Services-Enabled Service-Oriented Architecture

The SOA is an architectural style for building software applications that use services available in a network such as the Web (Mahmoud, 2005). It promotes loose coupling between software components so that they can be reused. Applications in SOA are built based on services, which constitute implementations of well-defined business functionalities and can then be consumed by clients in different applications or business processes, enabling enterprises to leverage existing investments by allowing them to reuse existing applications and promise interoperability between heterogeneous applications and technologies. SOA-based applications are distributed multi-tier applications that have presentation, business logic, and persistence layers. Services are the building blocks of SOA applications. While any functionality can be made into a service, the challenge is to define a service interface that is at the right level of abstraction. Services should provide coarse-grained functionality. SOA is emerging as the premier integration and architecture framework in today’s complex and heterogeneous computing environment. Previous attempts did not enable open interoperable solutions, but relied on proprietary APIs and required a high degree of coordination between groups. SOA can help organizations streamline processes so that they can do business more efficiently and adapt to changing needs and competition, enabling the software as a service concept.

Web services, the preferred standards-based way to realize SOA, are designed to support interoperable machine-to-machine interaction over a network. This interoperability is gained through a set of Extensible Markup Language (XML)-based open standards. In specific, the Web services architecture (WSA) and the Web Services Interoperability Model (WS-I) comprising three emerging key technologies: such as Web Services Description Language (WSDL), Simple Object Access Protocol (SOAP), and UDDI. These standards provide a common approach for defining, publishing, and using Web services. The Web services interface is described in a machine-processable format (specifically WSDL). Other systems and Web services interact with the Web service in a manner prescribed by its description using SOAP-messages, typically conveyed using Hyper Text Transfer Protocol (HTTP) with an XML serialization in conjunction with other Web-related standards.

In the literature, the Web services are defined as:

1. “loosely coupled, reusable software components that semantically encapsulate discrete functionality and are distributed and programmatically accessible over standard Internet protocols,”

2. “a new breed of application, which are self-contained, self-describing, modular applications that can be published, located, and invoked across the Web. Web Services perform functions, which can be anything from simple request to complicated business processes.”
The typical business scenario (Kreger, 2001), invoking and benefiting from the Web services-oriented solutions, identifies as core element of the implementation of the Web service architecture the UDDI services registry that acts as an intermediary between Web services providers and requesters, storing and categorizing services in taxonomies (directory services) (see Figure 2). The service provider deploys Web services and defines their service description, representing its available services, applications, and system features and publishes them in the service registry. The service requester takes advantage of the search capabilities of the registry’s directory service, searches the registry trying to find the composed service required and uses it, binding with the service provider. The main entities identified in a Web services-based business scenario, the service registry, the supplier (service provider), and the client (service) requester, interact in three ways: (1) the service provider publishes (publish activity) the WSDL service description in the service registry in order to allow the requester to find it, (2) the service requester retrieves (discover activity) a service description directly or queries the service registry for the type of service required, and (3) the service requester invokes or initiates an interaction (invoke activity) with the service at run time using the binding details in the service description to locate, contact, and invoke the service.

Web services, in their current form of loosely bound collections of services, are more of an ad hoc solution that can be developed quickly and easily, published, discovered, and bound dynamically (Samtani & Sadhwani, 2001). Web service-enabled SOA encourages and supports the reuse of existing enterprise assets, for example, already developed services and applications and allows the creation and deployment of new services from the existing infrastructure of systems. In other words, the Web service-enabled SOA facilitates businesses to leverage existing investments by allowing them to reuse existing applications and promises interoperability between heterogeneous applications and technologies. SOA provides a level of flexibility that was not possible before (Mahmoud, 2005) in the sense that:

Figure 2. Web services architecture, models and standards
• The Web services are software components with well-defined interfaces that are implementation independent, separating completely the service interface from its implementation. The deployed Web services are used and consumed by clients (services requesters) that are not concerned with how these services will execute their requests.

• The Web services are self-contained (perform predetermined tasks) and loosely coupled (for independence).

• The Web services can be dynamically discovered.

• Composed services can be built from aggregates of preexisting Web services.

A few essential differences between traditional EAI solutions and Web services (Samtani & Sadhwani, 2001) are presented in Table 1.

Although, the Web services applied to specific EAI scenarios provide an abstraction and flexibility layer supporting SOA and simplifying the application integration, they are based on exclusively syntactical-oriented technologies, not defining formally the semantics of services interfaces and of the data structures of the messages Web services exchanges. The main reason resulting in the failure of the majority of EAI implementations (some articles even account for 70% of EAI projects as failure) is that the semantics of different systems have to be formally defined and integrated at one point. The lack of formal semantics regarding the applications and services to be integrated makes it difficult for software engineers and developers to manually interconnect heterogeneous applications, impeding the automation regarding application integration, data exchange, and complex services composition. Engineers integrating the enterprise application systems have to know the meaning of the low-level data structures in order to implement a semantically correct integration. No formal definition of the interface data exist (Bussler, 2003b), which implies that the knowledge of every developer of applications involved in the integration project is assumed to be consistent.

Therefore, the problem that still exists, which the traditional Web services technologies are weak to solve, refers to the formalization and the documentation of the semantics related to the interfaces and the data structures of the deployed Web services. By applying Semantic Web technologies to SOAs and deploying Semantic Web services so as to integrate various systems, the notion of Semantic Web services enables SOA is emerging, paving the way to the semi-automated semantic-based enterprise application integration.
Table 1. Traditional EAI and Web services: Identified differences

<table>
<thead>
<tr>
<th>Aspect</th>
<th>Traditional EAI vs. Web Service Enabled EAI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simplicity</td>
<td>Web Services are much simpler to design, develop, deploy, maintain, and use as compared to a typical, traditional EAI solution which may involve distributed technology such as DCOM and CORBA.</td>
</tr>
<tr>
<td>Reusability</td>
<td>Once the framework of deploying and using Web Services is ready, it is relatively easy to compose new, aggregated services, reuse the existing IT systems infrastructure and automate new business processes spanning across multiple applications.</td>
</tr>
<tr>
<td>Open Standards</td>
<td>Unlike proprietary, traditional EAI solutions, Web Services are based on open XML-based standards such as WSDL, UDDI, SOAP and this is probably the single most important factor that leads to the wide adoption of Web Services technologies. Web Services are built on existing and ubiquitous protocols eliminating the need for companies to invest in supporting new network protocols.</td>
</tr>
<tr>
<td>Flexibility</td>
<td>Traditional EAI solutions require endpoint-to-endpoint integration. Changes made at one end have to be propagated to the other end, making them very rigid and time consuming in nature. Web Services based integration is quite flexible, as it is built on loose coupling between the application publishing the services and the application using those services.</td>
</tr>
<tr>
<td>Cheap</td>
<td>Traditional EAI solutions, such as message brokers, are very expensive to implement. Web Services, in the future, may accomplish many of the same goals - cheaper and faster.</td>
</tr>
<tr>
<td>Scope</td>
<td>Traditional EAI solutions consider and treat applications as single entities, whereas Web Services allow companies to break down complex services into small independent logical units and build wrappers around them.</td>
</tr>
<tr>
<td>Efficiency</td>
<td>Web Services allow applications and services to be broken down into smaller logical components, which make the integration of applications easier as it is done on a granular basis.</td>
</tr>
<tr>
<td>Dynamic</td>
<td>Web Services provide a dynamic approach to integration by offering dynamic interfaces, whereas traditional EAI solutions are pretty much static in nature.</td>
</tr>
</tbody>
</table>

Semantic Web Services in EAI Scenarios

The Emerging Semantic Web Services

The long-term goal of the Web services effort is seamless interoperation among networked programs and devices. Once this is achieved, Web services can be seen as providing the infrastructure for universal plug-and-play and ubiquitous computing (Weiser, 1993). However, the main obstacle of achieving interoperability among
deployed Web services is that the technical and functional description (profile) of the services is based on semi-formal natural language descriptions, which are not formally defined, not allowing computers to understand and interpret the data to be exchanged among Web services. The Semantic Web initiative’s purpose is similar to that of the Web services (Preece & Decker, 2002): to make the Web machine processable rather than merely “human processable.” Thus, Web services are considered as an essential ingredient of the Semantic Web and benefit from the Semantic Web technologies. Key components of the Semantic Web technology are:

- a unified data model such as RDF,
- languages with well defined, formal semantics, built on RDF, such as the Web ontology language (OWL) DARPA agent markup language and ontology inference layer (DAML+OIL), and
- ontologies of standardized terminology for marking up Web resources, used by semantically enriched service level descriptions, such as OWL-S (former DAML-S, DAML-based Web service ontology).

Enriching Web services descriptions with formal defined semantics by introducing the notion of semantic markup, leading towards the Semantic Web services (see Figure 3), enables machine-interpretable profiles of services and applications, realizing the vision of dynamic and seamless integration. As this semantic markup is machine—processable and—interpretable, the developed semantic profiles of Web services can be exploited to automate the tasks of discovering Web services, executing them, composing them, and interoperating with them (McIlraith, Son, & Zeng, 2001), moving a step forward towards the implementation of intelligent, Semantic Web services.

The combination of Web services and Semantic Web technologies, resulting in the deployment of machine processable and, therefore, usable for automation Semantic Web services, supports and allows a set of essential automated services regarding the use of deployed Web services (McIlraith et al., 2001a; McIlraith et al., 2001b):

- automatic Web service discovery, involving automatic location Web services that provide a particular functionality and that adhere to requested properties expressed as a user goal,
- automatic Web service composition, involving dynamic combination and aggregation of several Web services to provide a given functionality,
- automatic Web service invocation, involving automatic execution of an identified Web service by an agent, and
automatic Web service interoperation within and across organizational boundaries.

These semantically enriched Web services-oriented features can constitute the ideal solution to integration problems, as they enable dynamic, scalable, and reusable cooperation between different systems and organizations. Table 2 summarizes the main improvements that the semantic markup resulted in Web services:

**Semantic Web Services Registries**

As presented in the first section, the Web services architecture involves three core entities: (1) the service provider (supplier), (2) the service requester (client), and (3) the business services registry serving as a business mediator. The Semantic Web services deploy a similar architectural schema, with the crucial difference that the service technical and functional descriptions are semantically enriched with concepts defined in reference ontologies. However, current widely—known and—used service registries (i.e., UDDI and ebXML registry) specifications and implementations do not support the effective handling of semantic profiles of Web services, and a number of research activities have taken place, recently, trying to semantically enrich the standardized service registries. Their common goal has focused on the capability of registries to store and publish semantic data, so as to facilitate the semantic-based description of Web services, the ontology-based categorization and discovery of Web services, and, therefore, the semantic integration of business services and applications.

In specific, Moreau, Miles, Papay, Decker, and Payne (2003) present an approach and implementation for service registration and discovery that uses an RDF triple store to express semantic service descriptions and other task/user-specific metadata, using a mechanism for attaching structured and unstructured metadata. The result
is an extremely flexible service registry that can be the basis of a sophisticated semantically enhanced service discovery engine. This solution extends service descriptions using RDF and changes UDDI APIs for support of semantic search. Moreover, Pokraev, Koolwaaij, and Wibbels (2003) present the design and implementation of an enhanced UDDI server, capable of storage, matching, and retrieval of semantically rich service profiles that contain contextual information, mapping DAML-S to UDDI publish message and introducing, with their approach, additional elements such as a matchmaker, an ontology repository, and a proxy API to invoke UDDI APIs. The approach of Pokraev et al. (2003) does not change the publish and inquiry interfaces of the UDDI. In addition, Paolucci, Kawamura, Payne, and Sycara (2002) show how DAML-S service profiles, which describe service capabilities within DAML-S, can be mapped into UDDI records and how the encoded information can be mapped within the UDDI registry to perform semantic matching. This work proposes semantic search based on an externally created and operated matchmaker, as the semantic data are stored outside of the UDDI registry, while the mapping is implemented with links from the UDDI tModel to the semantic profile of the Web service. Finally, Srinivasan, Paolucci, and Sycara (2005) base the discovery mechanism on OWL-S. OWL-S allows to semantically describe Web services in terms of capabilities offered and to perform logic inference to match the capabilities requested with the capabilities offered. Srinivasan et al. (2005) propose OWL-S/UDDI matchmaker that combines the better of two technologies.

As shown previously, current technologies and research efforts, towards the realization of semantic-enriched services registry, use current UDDI implementation and try to extend their functionalities with semantic-based capabilities, introducing external matchmakers and mapping techniques. We claim that a pure semantic-based implementation of the UDDI specification, called pure semantic registry, provides a flexible, extendable core architectural component to allow the deployment and

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Existing Web Services</th>
<th>Semantic Web Services</th>
</tr>
</thead>
<tbody>
<tr>
<td>Services</td>
<td>Simple</td>
<td>Composable</td>
</tr>
<tr>
<td>Requestor</td>
<td>Human (developer)</td>
<td>Agent</td>
</tr>
<tr>
<td>Provider</td>
<td>Registration</td>
<td>No registration</td>
</tr>
<tr>
<td>Mediator</td>
<td>Key Player</td>
<td>Facilitator</td>
</tr>
<tr>
<td>Description</td>
<td>Taxonomy</td>
<td>Ontology</td>
</tr>
<tr>
<td>Descriptiveness</td>
<td>Closed world</td>
<td>Open world</td>
</tr>
<tr>
<td>Representation</td>
<td>Syntax-based</td>
<td>Semantics-based</td>
</tr>
</tbody>
</table>
business exploitation of Semantic Web services. The implementation of the PSR involves the design and development of a semantic-based repository and an embedded RDF-based reasoning engine. The PSR enables and supports the storage, administration, and handling of the deployed Semantic Web services and their profiles in a unique semantic repository. The semantic service profiles are annotated by using internally store domain ontologies facilitating, thus, the ontology-based categorization of services. Finally, the semantic registry benefits from its powerful RDF-based query and inference engine to support effective service discovery and composition.

**FUSION Conceptual Framework**

**FUSION: Towards the Business Intelligent Semantic SOA**

The FUSION solution is an integration framework that facilitates the integration of heterogeneous enterprise applications that exist in the same organization or in different organizations. The design of the FUSION approach has been based on a layer-oriented architecture (see Figure 4), using several structural components and preexisting technologies (Web services, semantics, services registry, etc.) benefiting from the typical advantages of each technology. This innovative, structured compilation of technologies and EAI techniques reduces the integration obstacles, which each technology when applied to EAI scenarios could face, enabling the intelligent integration of business services.

*Figure 4. Layer-oriented EAI architecture*
In specific, FUSION framework involves:

- A Web services infrastructure, which provides an initial interoperable capability based on Web services interface and communication integration, serving as a common deployment basis for all the enterprise applications and business services. As the Web services infrastructure applies the notion of SOA to the proposed framework, FUSION basis constitutes a pragmatic, applied SOA architecture.

- A semantic enrichment layer, which adds semantics to the technical and functional descriptions of the Web services, making the ontology-annotated Web services understandable and profiles machine interpretable. The semantic enrichment layer extends the notion of SOA with formal, well-defined semantics, moving towards a semantically enriched SOA.

- A semantic registry that constitutes an implementation of the latest UDDI specification based on Semantic Web technologies, supporting and semantically extending the main functionalities of service registries (i.e., UDDI and ebXML registries): the storage, categorization and discovery of the deployed business Web services. The FUSION semantic registry does not proposes a new registry architecture and specification, but it constitutes an alternative of the implementation that benefits from the intelligent ontology-based categorization, the strong RDF-based query language and inference engine.

- A business process layer facilitating the design and execution of Web services processes and workflows. The designed workflows invoke the business services stored in the semantic registry, retrieving them by using the semantic-based services of the registry. The interaction of the process design and execution environment with the service registry facilitates the automatic service discovery, composition, and invocation, supporting the interoperability among previous incompatible enterprise applications.

- A business scenarios and rules layer that defines and models, using formal ontologies that conceptualize e-business and B2B transactions, typical business scenarios occurring within companies and/or across collaborating enterprises. The formal business rules are transformed into parameterized workflow models, and are executed within the business process layer.

The upper two business-oriented layers, the business process layer and the business scenarios and rules layer adds business intelligence in the applied SOA, allowing the automated composition and orchestration of the deployed Web services, and supporting the automatic integration of business services. Apart from the aforementioned layers, the FUSION framework involves an ontology-based layer, which interacts with most of the rest of the integration layers. FUSION ontologies, which formalize
the concepts, the relations, and the events existing in an e-business environment, are separated in three main ontologies:

- The **business data ontology** defines the basic business data types and relations used in business services and transactions. The business data ontology is taken into consideration in the semantic enrichment of the deployed Web services, so as to define formally the data structure of the SOAP messages exchanged during a business transaction.

- The **business service ontology** conceptualizes the functionality of a given application that is used to annotate the functional profiles of Web services (during the semantic enrichment phase).

- The **business scenarios ontology** models the business rules identified, by business analysts and consultants during the business scenarios phase, in typical inter- and intra-organizational business scenarios. The ontology-based business rules defined are used in the business processes design to enable the composition of complex, aggregated Web services.

The next sections present in detail the FUSION conceptual framework, specify the several integration layers required for realizing business intelligent semantic SOA applied to inter- and intra-organizational and/or enterprise EAI scenarios, analyze how FUSION ontologies extends the OWL-S upper ontology concepts, and define the OWL-S representation of services.

**FUSION Integration Layers**

*Web Services Infrastructure and Semantic Enrichment Layer*

The conceptual architecture of the FUSION integration approach is based on a Web services infrastructure (see Figure 5). The, so-called, *Web service-enabled SOA infrastructure* allows the deployment of Web service software instances of each business applications and services, respectively, so as to provide a first integration layer, regarding the interfacing (WSDL) and communication (SOAP) of initially incompatible business applications.

Although, this first layer of abstraction, involving WSDL interfaces, provides a universal standards-based, highly flexible and adaptable implementation of business applications integration (Haller et al., 2005), the problem of documenting and understanding the semantics of these interfaces not only still exists, but it becomes a crucial issue to deal with. The significance of interpreting semantics in a machine understandable way arises from the continuously increasing average amount of
Web services that are stored in typical UDDI registries used in the Web service-enabled SOA approach, which makes it difficult for the developer and/or software engineer to manually integrate and put together the suitable Web services. That is why FUSION framework contains a second integration layer (see Figure 5) that "adds formal and well-defined business data and services functionality semantics in the Web services descriptions and interfaces," enlarging the notion of SOA and Web services applying common reference business ontologies.

This second integration layer supports the semantic enrichment of the Web services descriptions (WSDL files) taking into account two basic facets. Firstly, we should provide a formal description of the functionality of the Web service in order to facilitate efficient categorization and discovery of Web services. Therefore, the business service ontology is needed to identify the events that could occur in an e-business and/or B2B environment and to organize the business logic of this domain, creating an ontology-based dictionary conceptualizing functionality aspects of potential services of the e-business domain.

As real-life business services contain several and quite complex parameters and structures, we have recognized the need of developing the business data ontology formalizing the types of data contained in WSDL interfaces as well as the structure of the information that Web services exchange through SOAP messages. So, the FUSION second integration layer provides the mechanism, the graphical interface, and the common-reference business ontologies, to semantically annotate the Web services descriptions.

Figure 5. FUSION (Semantic) Web services-enabled SOA infrastructure
services profiles using the appropriate functionality and data concepts, and to create semantically enriched OWL-S descriptions of the Web services software instances, applying and leveraging the use of the *Semantic Web services in service-oriented architecture* deployed to business environments.

**Semantic Business Services Registry**

Once the Web services instances are deployed and their OWL-S semantic profiles are created, they should be categorized and published in business service registries in order to allow users (i.e., agents and humans) to discover, compose, and use, on demand, the services published there. As the most common service registries (i.e., UDDI and ebXML registries) do not support the storage and maintenance of ontologies and/or semantic profiles—Internally to the registry, methods have been developed to associate the set of semantics that characterizes a Web service with the service advertised through the business registry. A common drawback identified to all the existing techniques, trying to add semantics or semantically enrich predefined service registries, is that the reference ontologies and the semantic profiles of the Web service instances are stored externally to the registry, using informal, complex mapping tables and association rules to support the basic UDDI and ebXML registries services, they fail to embed effectively the dynamic and flexible Semantic Web technologies in the main services powered by such registries: categorization and discovery of Web services.

The FUSION approach has studied the methodologies and the lessons learned by research efforts focusing on the semantic enrichment of formal service registries and tries a different and innovative orientation. As the FUSION approach seeks to benefit more from the emerging Semantic Web technologies and standards, it moves towards the implementation of a “pure” *FUSION semantic registry*, based on a full functional RDF semantic repository (see Figure 6). FUSION approach develops a “thin-UDDI” API, internally to the semantic registry, to realize the basic set of functions of the traditional registries. In order for the proposed approach to be fully compliant with the dominant standards of the e-business domain (i.e., UDDI), FUSION transforms the XSD Schema of the latest UDDI specification in a RDF-Schema stored in the developed RDF repository, so as to preserve the widely known informational and relational infrastructure of the UDDI registry and to take advantage of its well-defined internal structure. This implementation benefits from the new possibilities provided by the RDQL query language when combined with the reasoning and inference engine of the RDF repository facilitates. Therefore, the FUSION semantic registry supports the storage and lifecycle management of RDF files and reference ontologies, internally, while it uses the query language and the inference engine provided to enable categorization and discovery services based on well-defined (formal) common semantics.
Furthermore, an upper layer of abstraction is needed in FUSION approach to move the EAI efforts, which follows the SOA and Web services architectures, a step forward towards the vision of the intelligent Web services and the business intelligent semantic SOA. This “ultimate” integration layer invokes the use of business process-driven workflows and modeling, taking into account and analyzing the most typical e-business and/or B2B scenarios, so as to design workflows that model the behavior of the selected business services in a business process interaction.

The intelligent SOA allows the experience and knowledge of business consultants and experts to be conceptualized and embedded to typical business scenarios, facilitating the formal modeling and execution of business processes using the Business Process Execution Language for Web Services (BPEL4WS) workflow modeling language. While the business consultants develop and model the desirable business scenarios, they define the Web services required by referring to the functionality aspects of services and using the common reference business services ontology. As this service functionality-oriented ontology is also used to annotate, characterize, and categorize the deployed Web service in the common semantic registry, the execution defined workflow models realizes the automated composition of intelligent Web services and the orchestration of flexible, complex business services.
There have been a number of efforts to add semantics to the discovery process of Web services. An upper ontology for services has already been developed and presented to the Semantic Web services project of the DAML program, called OWL-S (formerly DAML-S). OWL-S upper service ontology provides three essential types of knowledge about a service, each characterized by the question it answers:

- What does the service provide for prospective clients? The answer to this question is given in the “profile,” which is used to advertise the service. To capture this perspective, each instance of the class Service presents a ServiceProfile (see Figure 7).

- How is it used? The answer to this question is given in the “process model.” This perspective is captured by the ServiceModel class. Instances of the class Service use the property describedBy to refer to the service’s ServiceModel.

- How does one interact with it? The answer to this question is given in the “grounding.” Grounding provides the needed details about transport protocols. Instances of the class Service have a supports property referring to a Service-Grounding.

Figure 7. OWL-S service profile classes and properties
Generally speaking, the service profile provides the information needed for an agent to discover a service, while the service model and service grounding, taken together, provide enough information for an agent to make use of a service, once found.

The grounding concept in the OWL-S ontology provides information about how to access (invoke) the service, that is, details on the protocol, message formats, serialization, transport, and so forth. It is viewed as a mapping from an abstract to a concrete specification of those service description elements that are required for interacting with the service. OWL-S only defines such grounding for WSDL and SOAP (see Figure 8), although additional groundings can be defined. A summary of the automation support each upper level concept (or its subconcepts) of the OWL-S ontology is intended to cover is given in Table 3.

Table 3. Purpose of OWL-S upper level concepts

<table>
<thead>
<tr>
<th>Upper level concept</th>
<th>Automation support</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Profile</strong></td>
<td>• Discovery</td>
</tr>
<tr>
<td><strong>Model</strong></td>
<td>• Planning</td>
</tr>
<tr>
<td></td>
<td>• Composition</td>
</tr>
<tr>
<td></td>
<td>• Interoperation</td>
</tr>
<tr>
<td></td>
<td>• Execution monitoring</td>
</tr>
<tr>
<td><strong>Grounding</strong></td>
<td>• Invocation</td>
</tr>
</tbody>
</table>
Business-Oriented OWL-S Extension for Describing Web Services

In the complicated business services, the service profile should provide a clear description of the functionality of the service to be used, while the service model involves retrieving the suitable Web service and the service grounding the way the object is exchanged. As the OWL-S ontology provides a high abstraction layer for semantic description of Web services, a business-oriented extension of OWL-S service profile is needed (see Figure 9) to provide the ontology-based infrastructure enabling the semantic description of business services concerning three main aspects: (1) the business service provider entity, (2) the functionality of the Web service, and (3) the data types that the Web service exchanges.

This business-oriented OWL-S extension, called e-business and B2B ontology, provides the necessary semantics, concepts, classes, and interrelations, to characterize the Web services deployed by annotating the OWL-S profiles of services with formal, well-defined semantics.

FUSION Ontologies

For the realization of the business services ontology-based infrastructure that is presented in the paragraph, we have developed three interconnected ontologies, called the FUSION ontologies, that describe the various entities and components
that participate in business transactions. The FUSION ontologies serve the objective of making the technical realization as declarative as possible.

The FUSION ontologies constitute the cornerstone for the semantic description and modeling of business-oriented web services. The core objective of these business ontologies is to facilitate efficient business collaboration and interconnection between heterogeneous, incompatible services supporting the semantic fusion of service-oriented business applications that exist within an enterprise or in several collaborating companies.

The FUSION ontologies conceptualize the identified attributes, concepts, and their relationships of the service-oriented businesses applications and will be developed in three layers, each of them referring to a significant business entity—aspect: the service provider, the service functionality, and the services data types. This multi-layer architecture of FUSION ontologies provides a rich representation of service-oriented business applications, captures the significant requirements of both services functionality and data, supports efficient representation of services in intra- and inter-organizational level, and provides a flexible structure that could be easily refined and updated. The ontologies define:

- the basic description of the functionality that the business services provides to the end user (functional semantics) in order to capture the (semi-) formal representation of the functional capabilities of Web services in order to support the semantic-based discovery and automated composition of Web services, annotating the operations of services software instances as well as providing preconditions and effects—the business service ontology provides this type of information;
- the data types and relevant semantics required for representing the message structures and information that the Web services exchange (data/information semantics), capturing the (semi-) formal definition of data in input and output messages of a Web service, supporting discovery and interoperability by annotating input and output data of Web services using data-oriented ontologies—this information is specified in the business data ontology;
- the processes and scenarios identified in typical intra- and inter-organizational business transactions using a rule-based modeling approach (process and execution semantics), facilitating the automated composition and orchestration of complex Web services and workflows—this information is formally defined by the business scenarios ontology; and
- the categorization of the business entities that provide the deployed Web service software instances—this information is provided by the service provider ontology.
During the development of the FUSION ontologies, we have taken into consideration and examined already available ontologies and e-business standards. As a result, we have reused and built on already established and widely used domain knowledge, eliminating the danger of “reinventing the wheel.” So, we have based on two dominants XML-based business standards: ebXML (the Core Components Technical Specification and the Catalog of Common Business Processes) and RosettaNet (the Technical Dictionary and the Business Dictionary) defining a list of terms which can be used in business documents, as well as in other formal business vocabularies and taxonomies.

**FUSION Technical Implementation**

FUSION architecture is in line with the applied SOA architecture targeting smooth integration and dynamic service creation of services related with an ERP and a CRM system. Consequently the basis of the architecture is the ERP and the CRM software components. *FUSION adoption guideline* requires the existence of:

- a standard set of *exported Web services* that facilitate the software’s functionality. These Web services will be used for dynamic service creation during a complex service composition;
- a functional ontology, which is a domain specific ontology used for the semantic annotation of exported Web services; and
- An annotation procedure that aims at the semantically enrichment of Web services’ description.

*Figure 10. FUSION technical architecture overview*
**FUSION Architecture**

An overview of FUSION architecture is presented in Figure 10.

As mentioned previously, the elementary component in a SOA approach is Web services, since Web services provide a standard means of interoperating between different software applications running on a variety of platforms and/or frameworks. Web services are characterized by their interoperability and extensibility as well as their machine-processable descriptions thanks to the use of XML, and they can then be combined in a loosely coupled way in order to achieve complex operations. Consequently the first step of the FUSION adoption guideline is the provision of simple services derived from ERP and CRM functionality (domain specific functionality). This is an extremely crucial task since simple services can interact with each other in order to deliver sophisticated added-value services. However it is not a trivial task because SOA is a complete overhaul impacting how systems are analyzed, designed, built, integrated, and managed.

The next step is the semantic annotation of exported Web services and more specifically the semantic annotation of their WSDL file. As mentioned previously, WSDL is an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information. The operations and messages are described abstractly, and then bound to a concrete network protocol and message format to define an endpoint. Related concrete endpoints are combined into abstract endpoints (services). WSDL is extensible to allow description of endpoints and their messages regardless of what message formats or network protocols are used to communicate, however, the only bindings described in this document describe how to use WSDL in conjunction with SOAP 1.1, HTTP GET/POST, and MIME.

The cornerstone of FUSION architecture is, as expected, the *enterprise application server* which encapsulates the following modules:

- *semantic registry*, which is a variation of a classic Web services registry used for service discovery, and
- a *business process execution engine*, which executes Business Process Execution Language (BPEL) scenarios.

**Semantic Registry**

The extension of traditional Web services to Semantic Web services raises the necessity of semantic support in current Web services registries. A lot of effort has been put
into this field. Research that has been conducted with the aim of extending registries so they could support semantic discovery can be classified into two groups:

- those who extend legacy Web services standards by adding semantic annotation to reinforce the discovery function in registries, and
- those who preserve semantic advertisements into legacy registries by mapping semantic information into the registry information model.

FUSION approach aims to tackle this issue in a more unified way through the implementation of a PSR. PSR is a variation of a classic registry (UDDI, ebXML) that can store additional semantic metadata that accompany the Web service description model. PSR handles ebXML v.2.5 and UDDI v.3. At first all the entries of each registry are converted into OWL-S ontologies with additional classes. The persistence model of PSR is not based in a database but in an integrated ontology. Service discovery within the ontology is made using RDQL queries. The semantic registry utilizes Jena\(^\text{10}\) for storage and discovery. Jena is a Java framework for writing Semantic Web applications developed under HP Labs Semantic Web Programme. It features:

- statement-centric methods for manipulating an RDF model as a set of RDF triples,
- resource-centric methods for manipulating an RDF model as a set of resources with properties,
- cascading method calls for more convenient programming,
- built in support for RDF containers—bag, alt, and seq,
- enhanced resources—the application can extend the behavior of resources,
- integrated parsers and writers for RDF/XML (ARP), N3, and N-TRIPLES, and
- support for typed literals.

**BPEL Engine**

Since many organizations are moving from an object-oriented paradigm for managing business processes toward a service-oriented approach, services are becoming the fundamental elements of application development. At the same time, BPEL has become the de facto standard for orchestrating these services and managing flawless execution of business process. The confluence of these trends is presenting some
interesting opportunities for more flexible, cost-effective management of business processes.

ERP and CRM business processes contain multiple decision points. At these decision points, certain criteria are evaluated. Based on these criteria or business rules, business processes change their behavior. In essence, these business rules drive the business process. Frequently, these rules are embedded within the business process itself or inside custom Java code, which can cause several problems such as:

- Business rules change more often than the processes themselves, but changing and managing embedded business rules is a complex task beyond the abilities of most business analysts. Thus, as business rules change, programmers often have to commit expensive time to this task.
- Most organizations lack a central rules repository. Consequently, any organization-wide change in policy cannot be applied across all business processes.
- Business processes cannot reuse rules. Hence, IT personnel end up designing rules for each and every process, often leading to inconsistency or redundancy.

The best way to avoid these problems is to use a rules engine to separate business processes from business rules. In this approach, rules are exposed as services and BPEL processes leverage these services by querying the engine when they reach decision points. This approach is much more flexible—Instead of coding rules in programming languages or inside a process, rules can be manipulated graphically. Business users with tools can write rules themselves and make post-deployment rule changes without IT assistance. With business users doing most of the updates and enhancements, maintenance costs can be reduced substantially. Consequently, rule engines and BPEL are complementary technologies.

It is rather important to delineate rules from processes. Hence, a major decision in FUSION architecture is how to implement business policies, business processes, and supporting business logic. Business logic is spread across three different layers of the IT infrastructure: (1) business process, (2) Web services, and (3) rules (see Figure 11).

**Business Process Layer**

This layer is responsible for managing the overall execution of the business process. These business processes, implemented using BPEL, can be long running, transactional, and persistent. The BPEL engine supports audit and instrumentation of workflow and thus is well suited for:
- separating less volatile workflow steps from more volatile business rules,
- implementing line-of-business processes,
- implementing process flows requiring compensation,
- supporting large-scale instantiation of process flows,
- designing process flows that need auditing, and
- orchestrating heterogeneous technologies such as connectors, Web services, and Web Services Invocation Framework (WSIF)-enabled logic.

**Semantic Web Services Layer**

The Web services layer exposes the existing application layer functionality as services. Multiple business processes can then reuse these services, thereby fulfilling the promise of a SOA.

Web services implement functional and domain logic. Functional methods are typically stateless and medium grained. Web services may, for example, contain utility methods, entity operations, and inquiry methods for system data. Web services can be implemented using multiple technologies and hide differences among implementation platforms. Thus, this layer is well suited for:
implementing medium-grained methods for a particular entity/domain area,
integrating legacy code/third-party tools, and
encapsulating logic, custom code, and implementation from the application layer.

**Rules Layer**

The rule engine is typically the home for complex logic that involves a number of interdependencies between entities and order-dependent logic calculation. Extracting business rules as a separate entity from business process leads to better decoupling of the system, which, in consequence, increases maintainability.

Rules engines allow for evaluation of rules sets in parallel and in a sequential order. In addition, rules have the ability to evaluate the values of input and intermediate data and determine if a rule should be fired. This modular design provides a simpler and more maintainable solution than traditional Java procedural code.

Furthermore, rules are declarative and allow high-level graphical user interface (GUI) editing by business analysts. Modern rule engines execute extremely quickly and provide built-in audit logging. The typical traits of a rules layer are as follows:

- contains coupled and complex logic,
- supports efficient business logic evaluation using parallel execution,
- contains complex return structure built from multiple business rule evaluations,
- allows for translation of domain logic into simple rules, and
- implements highly volatile business policy.

Because rules are exposed as services in the Web services layer, they can be reused across all inter-enterprise applications, making the development of new applications and integrations easier.

In the scope of FUSION approach BPEL4WS has been used. BPEL4WS provides a language for the formal specification of business processes and business interaction protocols. By doing so, it extends the Web services interaction model and enables it to support business transactions. BPEL4WS defines an interoperable integration model that should facilitate the expansion of automated process integration in both the intra-corporate and the B2B spaces. IBM BPWS4J has been utilized in the scope of FUSION solution. BPWS4J includes a platform upon which can be executed business processes written using the BPEL4WS and a tool that validates
BPEL4WS documents. Additionally, the enterprise application server includes a scenario repository that stores already existing BPEL scenarios for future use.

**FUSION Adoption: Integration Scenario and Applying Methodology**

**Typical Integration Scenario: Multinational, Franchising Firms**

A typical use case scenario, applying FUSION framework to solve EAI problems, refers to multi-national, franchising firms and is presented in the following section. Multi-national, franchising firms constitute a typical integration case, because of the fact that they involve several, geographically distributed legacy systems that need to be integrated at one point so as to facilitate the exchange of crucial business information among the networked franchising companies. As national systems work in isolation, any business interaction between headquarters is done currently, by mail, phone, or fax. Today, most of the steps in international workflows require human participation and batch data exchange to complete. For example, phone calls and human conversations are instantiated to carry out simple product availability requests and mails containing financial reports are exchanged for the purpose of financial auditing.

Humans, by making implicit interpretations of exchanged information, can reach a common understanding about things. Machines, on the contrary, require explicit and formal information interpretations in order to communicate. But, the company has concluded that manual execution of activities is expensive, while it does not allow jobs to repeat as often as needed. Human conversations and batch data exchanges are point-to-point interactions restricted to proprietary information structures. Even a fully automated point-to-point connection requires specific meanings and tightly bounded ends, which implies large volumes of implementation effort.

**Franchising Firms Application Scenario**

Product, inventory, demand, and financial concepts must have consistent meanings throughout the national headquarters network. For example, product classifications will keep a unique identity and a set of well-defined properties for each product across the enterprise. Once a common repository of semantics has been established,
Web services can be formally described by using common meanings from that pool. Services can then be published in registries public to all national headquarters, thereby becoming available for process composition. Semantic description and publishing of Web services deliver interoperable business services, which mean that services will exhibit consistent accessibility to any business process composite that wish to use it. Both stock management and purchase management processes may use a service that returns product stock levels in sibling headquarters and discovering and binding to that service will execute identically. Business operations planned for reengineering should be modeled from scratch and services recognized as parts should be described and published. Product availability and product stock level requests are business services that already exist in current stock management and purchase management processes.

By enabling national headquarters to publish loose-coupled, commonly accessed Web services the company becomes capable to compose highly automated business activities, avoiding thus human intervention. Services participating in a composite process of stock, purchase, or financial control are now selected from common pools (service registries). Therefore, no point-to-point connections are necessary, and the internals of the headquarters systems remain intact. Business processes are composed and executed at a higher semantic (abstract) level.

*Expected Results and Added Value: The Business Perspective*

The deployment of a business intelligent semantic service-oriented architecture to a multi-national, franchising firm, which requires several business transactions and information exchange, provides significant benefits to the firm, including:

- common access to all relevant information and functionality (interoperability), due to semantic networks and the common service registry (in place of “hard-wired” point-to-point connections),
- better quality of business services, due to standardization in service descriptions and publishing,
- business process reengineering (BPR) and analysis opportunities, due to changes that FUSION will bring in the very nature of business,
- faster responds to market changes, due to BPR flexibility,
- savings in resources, time, and money, as processes will be modeled and run automatically, and
- centralized management capabilities.
The FUSION solution intends to provide the national headquarters with a semantic service infrastructure, which will enable semantic service-oriented integration and interoperability, towards a vision of gradually incorporating all headquarters of a multi-national franchising firm into a virtual enterprise environment. The franchising firm should follow the adoption framework described next, in order to apply the FUSION integration solution to its enterprise system environment.

**A Methodology for Applying the FUSION Solution**

As described in the previous sections, FUSION solution allows the integration of heterogeneous enterprise applications that exist in the same organization or in different organizations. The FUSION solution involves the creation, administration, and deployment of Web services software instances of preselected features of the enterprise applications and the development of their semantic description (profile) based on the annotation of the technical descriptions of Web services with functionality concepts and semantics defined in the FUSION ontologies that serve as a common reference allowing the semantic integration of the business applications. The deployed Web services instances and their developed profiles will be stored.
and published at the business services registry (pure semantic registry [PSR]) that constitutes a semantic-based implementation of the UDDI specification and supports the categorization and discovery services of the PSR.

The step-oriented way we envision software engineers and business analysts of cooperating enterprises and organizations (service providers) to work with the FUSION solution (see Figure 12), in order to allow the semantic interoperability based on business intelligence among former incompatible business services and applications, is presented as follows:

- **Step 1. “As is analysis” of the pilot experiments.** This constitutes an in-depth analysis of the current situation of the service providers. The business analysts identifies the business systems and applications (e.g., legacy systems, ERP, CRM, SCM, etc.) existing within the environment of the service providers and selects the specific features and services of the existing business systems to be semantically integrated. The business analysts specifies both technically and functionally the selected business services.

- **Step 2. Deployment of Web service software instances.** The software engineers of the service provider company create and administrate Web services instances that realize the preselected features of the business applications.

- **Step 3. Web service semantic profile creation.** The business analysts identifies the concepts (e.g., product, contact, order) that are related to the deployed Web services and use well-defined concept models (business data and services ontologies) to enrich the technical description of the Web services instances.

- **Step 4. Semantic profiles publishing.** The software engineers register the semantically enriched functional and technical profiles of the provided business services on the PSR. The registered Web services are published at the, so-called, “yellow pages” of the registry, which support fully functional ontology-based categorization and discovery services.

- **Step 5. Business concepts analysis.** The business analysts identify the typical business scenarios involving the preselected enterprise applications. The business analyst defines formally the concepts and relations that exist within the identified scenarios and models these integration scenarios using a rule-based approach formalized in the developed business scenarios ontology.

- **Step 6. Services orchestration.** The software engineers design workflows that materialize the aforementioned identified business scenarios so as to support the semantic-driven orchestration of aggregated, complex compositions of Web services instances.
A service provider or a group of collaborating service providers should precede in the implementation of the activities described in these six phases in order to realize selected integration scenarios.

## Conclusion and Future Work

In this chapter, we have proposed a semantic integration framework, called FUSION, based on Web services and Semantic Web technologies. Our proposed approach introduces the deployment of SE-SOBAs that enlarge the notion of SOA by using ontologies to describe data structures and messages passed through Web service interfaces. We have also proposed the development of a pure semantic-based implementation of the UDDI specification, called Pure Semantic Registry.

The combination of SE-SOBAs with the pure semantic-based registry and the rule-based formalization of business scenarios and processes constitute a business-driven semantic integration framework applied to intra- and inter-organizational integration scenarios. Moreover, we have specified the FUSION adoption framework that constitutes a light, concrete methodology that supports enterprises and organizations to apply the FUSION integration solution to their enterprise system environment, as well as a typical integration scenario that uses the case of multinational, franchising firms.

The combination of Web services, Semantic Web technologies, and SOA results in the deployment of semantic SOA architectural framework, which is based on machine processable and, therefore, usable for automation semantic Web services, supporting a set of essential automated services regarding the use of the deployed SE-SOBAs: (1) automatic SE-SOBAs discovery, automatic complex, (2) aggregated SE-SOBAs composition, (3) automatic SE-SOBAs invocation (execution), and (4) automatic SE-SOBAs interoperation within and across organizational boundaries. The proposed semantic SOA framework, FUSION, enables the formalization and the documentation of the semantics related to the interfaces and the data structures of the deployed Web services, a capability that could not be supported by the current Web services-enabled SOA and technologies.

As the functional and technical FUSION architecture is already well specified and defined, the basic technical, structural components are being developed. However, a lot of work is still to be done towards the finalization of the integrated FUSION technical solution, its deployment in real enterprise scenarios, and the evaluation of the proposed semantic service-oriented architecture.
Acknowledgments

The work presented in this chapter constitutes the core conceptual and technical architecture and framework of a European Commission so-funded project, entitled FUSION. FUSION project is a specific targeted research project that focuses on semantic interoperability, enterprise application integration, and B2B process fusion. Led by SAP AG, the FUSION consortium consists of 14 partners from five European countries (Germany, Poland, Greece, Hungary, Bulgaria), including research institutes, technology providers, innovation transfer bodies, as well as end users.

References


---

**Endnotes**

2. http://www.w3.org/TR/ws-arch/
3. http://www.ws-i.org/Profiles/Basic/2003-05/BasicProfile-1.0-WGAD.htm
4. http://www.w3.org/TR/wsd1/
http://www.w3.org/TR/SOAP/

http://www.uddi.org/

The Stencil Group (www.stencilgroup.com/ideas_scope_200106wsdefined.html)


http://jena.sourceforge.net/

<table>
<thead>
<tr>
<th>Term</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Business Process</td>
<td>A collection of related structural activities that produce something of value to the organization, its stakeholders or its customers. The recipe for achieving a commercial result.</td>
</tr>
<tr>
<td>Business Process</td>
<td>Business process fusion is the transformation of business activities that is achieved by integrating the interfaces of previously autonomous business processes by pipelining different middleware technologies and enabling the effective (semi-)automated exchange of information between various systems within a company or between enterprises</td>
</tr>
<tr>
<td>CRM</td>
<td>Customer Relationship Management (CRM) enables organizations to better serve their customers through the introduction of reliable processes and procedures for interacting with those customers.</td>
</tr>
<tr>
<td>EAI</td>
<td>Enterprise Application Integration is the use of software and architectural principles to bring together (integrate) a set of enterprise computer applications. The goal of EAI is to integrate and streamline heterogeneous business processes across different applications and business units.</td>
</tr>
<tr>
<td>ERP</td>
<td>Enterprise resource planning system is a management information system that integrates and automates many of the business practices associated with the operations or production aspects of a company.</td>
</tr>
<tr>
<td>Service</td>
<td>Service is the non-material equivalent of a good provided to customers.</td>
</tr>
<tr>
<td>Se-SOBA</td>
<td>Semantically-enriched Service-Oriented Business Applications (SE-SOBA) - a set of independently running services communicating with each other in a loosely coupled message-based manner using ontologies and semantic web mark-up languages to describe data structures and messages passed through their web service interfaces</td>
</tr>
<tr>
<td>SOA</td>
<td>Service Oriented Architecture - a software architectural concept that defines the use of services, which communicate with each other involving simple data passing, to support the requirements of software users.</td>
</tr>
<tr>
<td>SOBA</td>
<td>Service Oriented Business Applications - a set of independently running services communicating with each other in a loosely coupled message-based manner</td>
</tr>
<tr>
<td>Web Service</td>
<td>Web service is a software system designed to support interoperable machine-to-machine interaction over a network</td>
</tr>
</tbody>
</table>