SEMANTIC DRIVEN ELASTIC SERVICE BASED FRAMEWORK FOR MODELING EAI

E. ANUPRIYA, N.BINDU THUSHARA
VIT UNIVERSITY, VELLORE, TAMILNADU, INDIA
eanupriya@vit.ac.in, binduthushara@yahoo.co.in

ABSTRACT

A number of frameworks has been proposed to model EAI using various approaches starting from HUB/SPOKE, ESB, SOA and recently using web services to deploy services more flexibly and economically. Providing services through web may correspond to semantic heterogeneity of services when mapping on to the legacy system making integration a challenging issue. This latter is not correctly addressed by today's solutions, which focus mainly on technical and syntactical integration. Dealing with the semantic aspect, which will certainly promote EAI by providing it more consistency and robustness, needs some appropriate principles such as the knowledge or ontology urbanization. The new proposed framework is an enhanced framework with ODSOI (Ontology-Driven Service-Oriented Integration) approach integrated with elastic service based framework to provide services based on semantics. It aims to favor the semantic service integration, precisely by providing it more appropriate knowledge structuring that can help any enterprise to correctly capture structure and master its semantics, which constitutes a big challenge for several enterprises that are in quest of more flexibility and manageability.

1. INTRODUCTION

Enterprise application integration is a business need to make diverse applications in an enterprise including partner systems to communicate to each other to achieve a business objective in a seamless reliable fashion irrespective of platform and geographical location of these applications.

Traditional Enterprise Application Integration (EAI) focuses on the integration of application interfaces by pipelining different middleware [6] technologies like message queuing or remote method invocations. Web service enabled Service-Oriented Architectures (SOAs) used in EAI were a step towards providing an abstraction layer for the involved interfaces by using the Web Service Description Language (WSDL) [4]. Numerous software companies have developed large, complex technologies specifically aimed at easing the problems that are inherent in application integration, especially in large enterprises with 10's or 100's of different legacy applications. This paper presents a comprehensive study on elastic service based framework for EAI [1] and ODSOI (Ontology-Driven Service-Oriented Integration) [2] and proposes a novel framework for modeling EAI based on semantics at service level.

2. APPROACHES TO EAI

There are two aspect of EAI: integration communication type and what is being integrated. There are three types of communication for EAI: SOA, Hub/Spoke, and ESB. Service oriented architecture (SOA) is approach to have software resources available and discoverable on network as services. Each service would achieve a predefined business objective and perform discrete units of work. The services are independent and do not depend on the context or state of the other services. Earlier SOA used COM or CORBA and recent SOA stress on web Services [2].

Service oriented architecture may or may not use web services but web services provide a simple way towards service oriented architecture. Hub/Spoke architecture uses a centralized broker (Hub) and adapters (Spoke) which connect applications to Hub [1]. Spoke connect to application and convert application data format to a format which Hub understands and vice versa. Hub on the other hand brokers all messages and takes care of content transformation/translation of the incoming message into a format the destination system understands and routing the message. In this mechanism, the central hub (message broker) contains the rules for connecting application together. The message exchange is based on a Hub and Spoke communication model, which enables applications to operate independently without forcing source applications to wait , receive the results of their requests .In addition, they provide rule processing capabilities, hosting business functions, message translation engines and bridges to many different platforms and applications (by using prebuilt AI adapters or existing APIs). Enterprise service bus (ESB) is a messaging backbone which does protocol conversion, message format transformation, routing, accept and deliver messages from various services and application which are linked to ESB on top of an Enterprise Messaging System. ESB implemented by technologies found in a category of middleware infrastructure products usually based on Web services standards that provides foundational services for more complex service-oriented architectures via an event-driven and XML-based messaging engine (the bus). Nowadays, these architectures are moved to

web services forcefully, due to the fitted technical solution which provides the required loose coupling to achieve such architectures .Things that are being integrated may include just the information, business processes, or both. Information integration focuses on providing an enterprise-wide integrated view of data entities to promote data integrity, shared semantics consistent usage, and rapid application development. Business process integration encompasses coordinating and executing a sequence of steps to accomplish a business task, each step may need to be executed by a different application.

3. ODSOI APPROACH

This section succinctly describes some important characteristics of our approach called ODSOI (Ontology-Driven Service-Oriented Integration) that rely on the use of both ontologies and Web Services technologies, and that aims to extend the state-of-the-art in EAI in order to address the semantic problem.

3.1 General principles

First of all, ODSOI approach is a solution to the information system integration problem[2]. This means that our approach addresses the heterogeneity problem by providing an *ontology-driven* integration solution that is based on ontology mediation.

Indeed, our approach is *service-oriented* (based on SOA) since it uses WSs for integrating EISs. The result architecture integration that we suggest is called ODSOA (Ontology-Driven Service-Oriented Architecture)[3]. This latter extends SOA with a semantic layer that aims to enhance service mediation in the context of EAI. Furthermore, our approach can support both *static* and *dynamic oriented integration*. This latter means that the binding services of target EISs can be performed at run time according to special binding rules, contrarily to a static integration solution where an integration scenario predefines every detail of potentially connectable EISs. The reasons of this choice are that the dynamic solution is sometimes more flexible and scalable form of integration than static one. In addition to this, the integration scenario becomes simpler and the configuration EIS interfaces can be changed and new systems can be added easily, even while the EAI system is running.

3.2 Global Architecture

The ODSOA concept provides a unified framework in order to integrate EISs. In this framework, three main types of services are defined: data-services, functional services and business-services. These different types can respectively address data, application and process integration. *Data-Services* (DS) [6] are services that expose data sources (EDS – Enterprise Data Sources) as services. *Functional-Services* (FS) are services that expose application systems, fundamentally functional systems (EAS – Enterprise Application Systems) as services. *Business-Services* (BS) are defined as the combination of the above services in order to expose business processes (EPS – Enterprise Process Systems) [5] as services. Our service typology enriches somewhat the topology proposed in and which is based on two types that are SaaS (Software-as-a-Service) and DaaS (Data-as-a- Service).

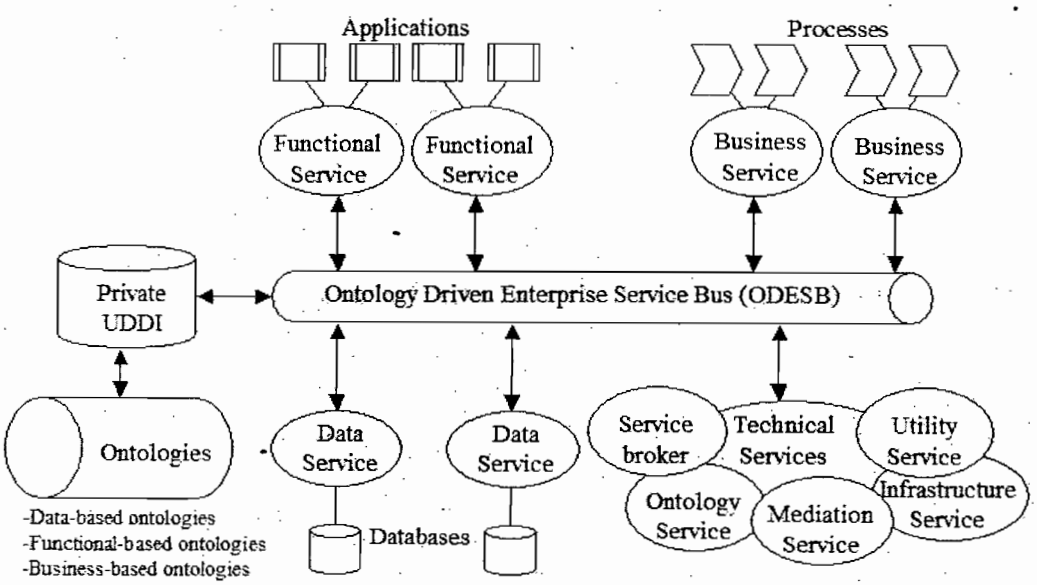

Fig. 1. Global view of ODSOA Architecture

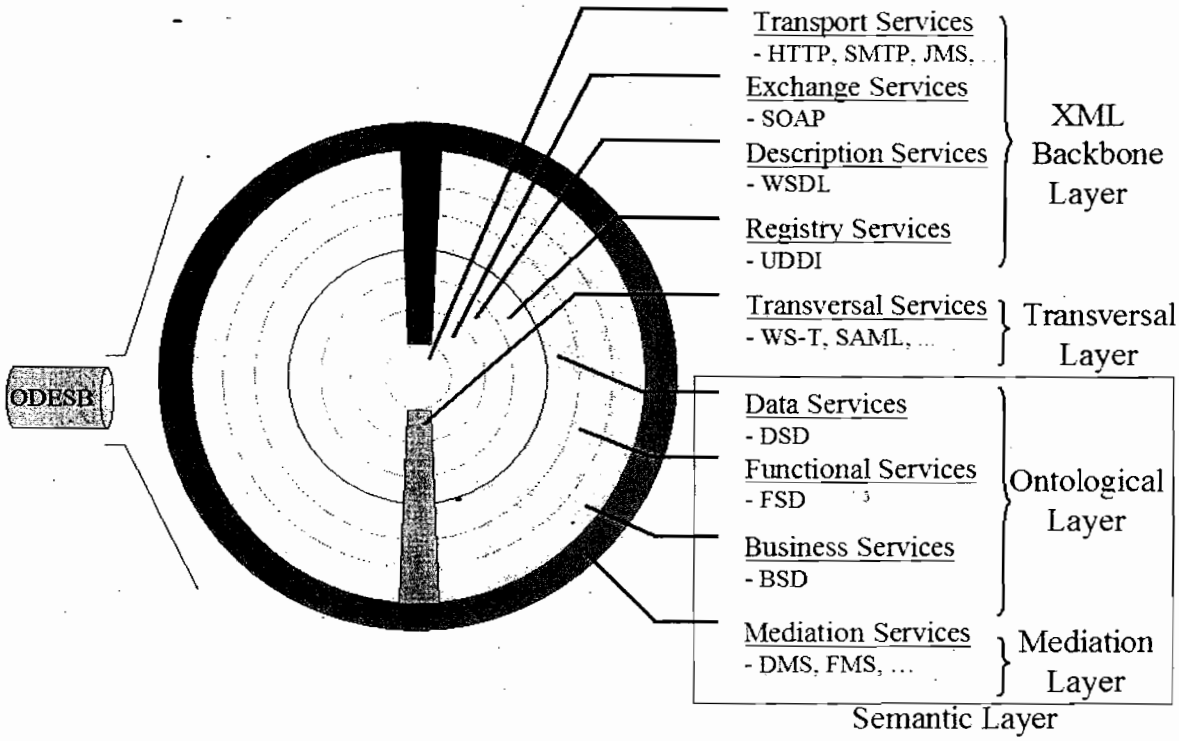

Fig. 2. A cross section of the integration bus (also called ODESB – Ontology-Driven Enterprise Service Bus)

Figure 1 which is a particular SOA recapitulates these important types of services. Indeed, there are, of course, some other important technical services that are mainly *service broker*, *ontology services*, *mediation services*, *infrastructure services* and *utility services*. Some of them will be described below.

A cross section of the integration bus (also called ODESB – Ontology-Driven Enterprise Service Bus)[2] (figure 2) shows many concentric standard layers such as *Transport layer*, *Exchange layer*, *Description layer*, *Registry layer* and *Transversal layer*. In addition to these standard and currently existing layers, we suggest to adopt, in a similar way as semantic web service concept , an other layer, called *semantic layer* and which contains two sub-layers that are *ontological layer* and *mediation layer*.

3.3 Overview of the Integration Process

ODSOI approach provides a uniform framework to integrate EISs at different levels: data, application and process integration level that are provided by three specific services or sub-frameworks that are respectively ODSODI (Ontology-Driven Service- Oriented Data Integration) framework, ODSOAI (Ontology-Driven Service-Oriented Application Integration) framework and ODSOBI (Ontology-Driven Service-Oriented Business Integration) framework.[2] Figure 3 illustrates the general principle of the three components of the global framework (ODSOI framework).

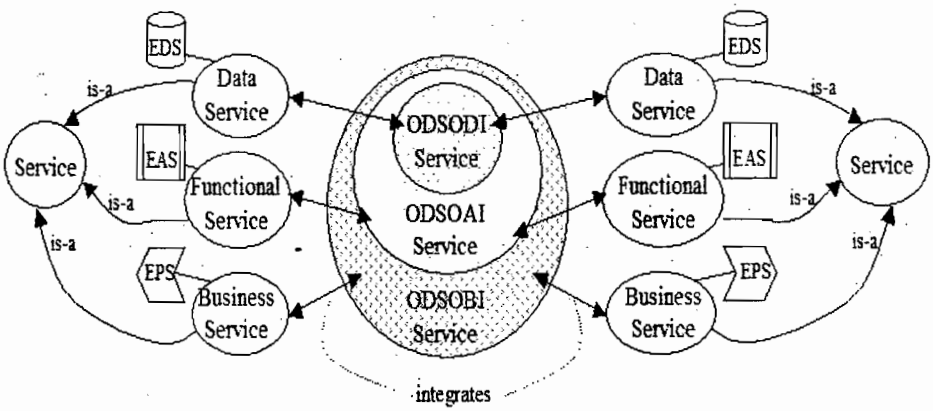

Fig. 3. General vision of ODSOI framework

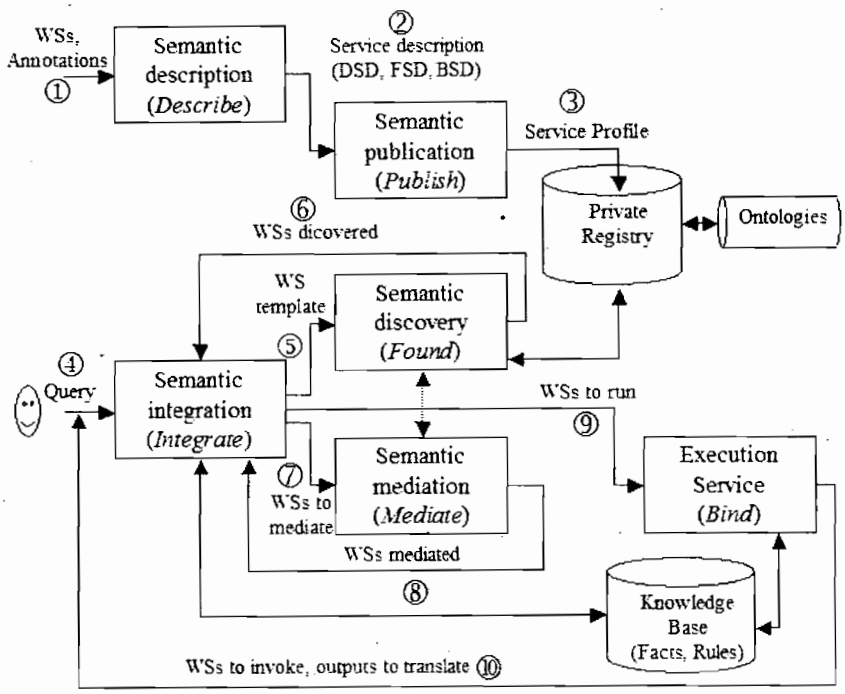

Fig. 4. Generic integration process

DSs can be integrated with the ODSODI framework by using data ontology mediation, whereas FSs can be integrated in similar way by using both data and functional ontology mediation. At last, in case of process integration, BSs can be integrated by using data, functional and business ontology mediation. These three sub-frameworks define stratification so that each level include its inferior levels. Since we can not detail the different sub-frameworks in this paper, so we just give a generic synopsis about the integration process.

The heart of integration process[2] is realized by mediation services with the help of a service broker. This latter is composed of different technical services such as integration service, discovery service, and execution service. The generic scenario of the integration process (figure 4) starts once the WSs have been described (semantic description) by using ontology services. After that, they are published (semantic publication) in enhanced private UDDI registries (by ontology services) and then they can be discovered (semantic discovery) by discovery service in order to carry out the realization of a given task modeled as a service template (that corresponds to a user or service query) by the integration service. The discovery service can use mediation service in order to perform the semantic matching. The invoked mediation services exploit a similarity function that calculate rapprochement between ontology concepts and then between the involved characteristics of services.

Once the desired WSs have been discovered, they are also mediated in order to resolve the semantic heterogeneity differences by the mediation service. Finally the mediated services are executed by the execution service and can invoke the integration service which can then perform another similar loop of the integration process. Furthermore, the above process includes dynamic integration which is made possible by some binding rules contained in the knowledge base. These rules exploit mainly the characteristics of services such as preconditions, effects and so on.

4. AN ARCHITECTURE FOR EAI

A framework is a template that can be repeatedly applied to different situations. It is quite common in business modeling and design, that some of the patterns of relationship and constraints occur repeatedly. Adaptability and flexibility have been the key considerations in evolving this architecture. The architecture is therefore general enough to suit a wide variety of scenarios. It suggests an open architecture and spans the four crucial areas of EAI communication, data, process and monitoring these through human intervention. An efficient, secure and robust Communication mechanism for enterprise applications to exchange information is the backbone of the framework. Data is the second pillar of EAI. The third important aspect of EAI is the co-ordination and sequence of the flow of information. The fourth requirement of EAI is the provision

of human intervention for controlling and configuring the way information is exchanged between applications. Various internal departments in the enterprise develop applications, which leverage the legacy systems; and these applications operate in an Intranet environment. Therefore, the objectives for this architecture are:

- Leverage legacy systems.
- Provide an integrated functional view of legacy systems as task-oriented, sharable atomic services.
- The services should be delivered via multiple delivery channels.
- Services should be configurable at multiple levels of granularity – service level, service group level, and system level.
- The integration architecture should promote platform independence and offer deployment options.
- It should be possible to programmatically use services with any industry standard programming languages.

4.1. Elastic Service Architecture Overview

The application integration architecture revolves around the envisaged framework .Figure 5 gives an overview of the components of the framework. As can be seen, the framework is fully service-based. External services are delivered via channels – Web services, message queue, CORBA, COM+ are some kind of services which can delivered but the external services is not restricted to this technology and the architecture can be expanded to include the desired services. Management of the internal services, which hosted by this architecture is achieved through web service and related technologies. The components of the framework are:

a) Adapter components, which Correspond to each of the delivery channels to convert consumer data format to a format which Integration Engine understands and vice versa.

b) Service Provider, which functions as an entry point into this architecture. It intercepts all service requests coming through the delivery channels, and hands them over to Process Orchestrator.

c) Process Orchestrator, which has total responsibility for processing a service request. Process Orchestrator has the requisite knowledge to process all services hosted by this architecture. Depending on the service request type, fulfilling the request May comprise several steps, and executing these steps in a Specific order.

d) Pre/Post processor is a very important component in this module. It can check the message integrity; find the appropriate application to handle the message, handling the exceptions etc. [8]

e) Repository is an XML and SOAP[8] based lookup service for translator to locate web services and their version information.

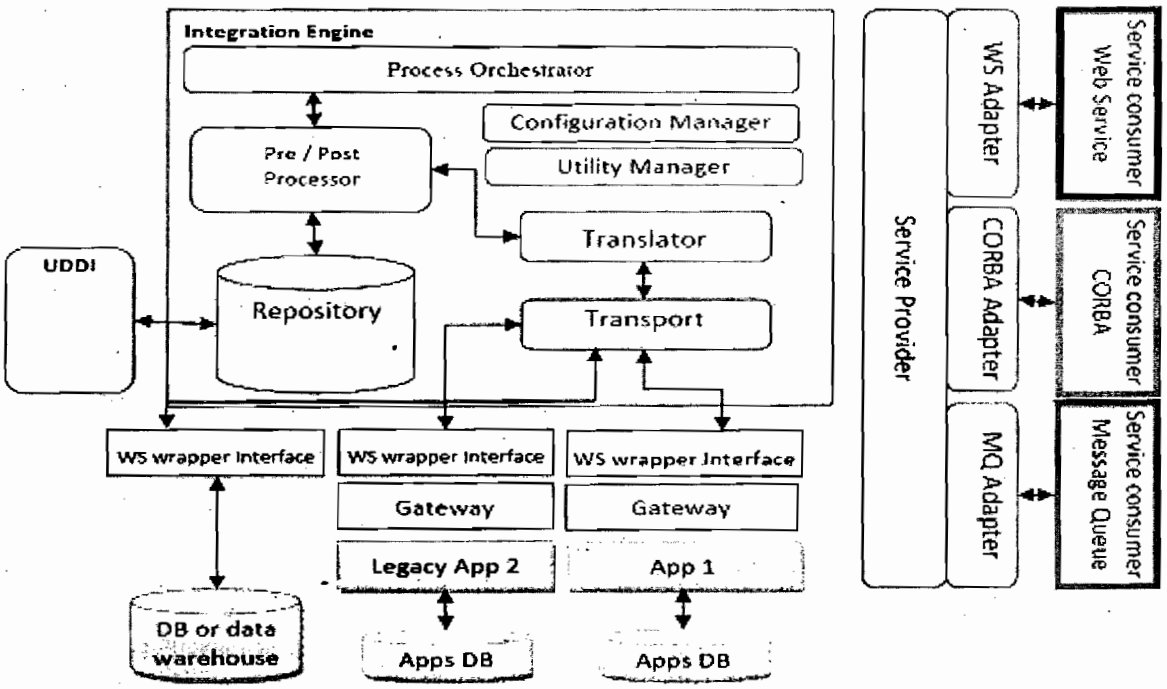

Fig. 5. Elastic service based architecture for EAI

This architecture in particular, depends excessively on efficient implementation of repository, because of the repeated use of the information stored in the repository. The registry should ideally use storage structures, which are more amenable to the querying mechanism of XML technologies. XSLT[8] processors, to be used in the Integration Engine would immensely benefit from such a storage structure . The repository can also be extended to lookup external databases like UDDI. e) Translator used to generate a suitable message structure to invoke the relevant API on the target Application. Although the overall message structure is pre-determined for a specific update service, the actual structure and length of an instantiated message depends on the parameter values supplied by the service invoker. Message structures are declaratively specified using repository. Thus, message structures for new services can be introduced without any changes to the source code of this architecture. f) Transport used to transport the message to the application using SOAP message format. This is the core function of Transport component. The transport configuration is done during the service core configuration. This component is capable of communicating with core services in FIFO mode; retry request mode; synchronous or asynchronous mode with data aggregation or a combination of them. g) Configuration Manager is used for configuring a service, a group of service or all the service as a unit. Configuration information includes database server type, connection string, database driver, location of XML files for system initialization, trace flags, and specifying what information goes into a trace file when the trace is enabled. Trace feature is used in debugging services. h) Utility Manager is a collection of utility classes for accessing process orchestrator and manipulating trace files.

4.2. Elasticity of architecture

The requirement was for a comprehensive integration infrastructure able to satisfy both immediate and anticipated requirements. Specific requirements included providing for application integration defining a multitier architecture to facilitate addition of access channels and applications, based on service orientated approach, and providing appropriate levels of system management. The approach taken was to develop an architecture based on the notion of a middleware bus. An analysis of the requirements in our case study revealed need for real-time interaction, with the option for global transactions across more than one system and database, including support of two-phase commit and compensation for long running transactions, in particular between Windows and Linux, Technology and product choices were COM+ and .Net in the Windows environment and EJB in Linux[1] .The other element is a set of common services, which are again implemented using web services [8] and are able to interface with applications in the various languages used. In order to combine automated processes and human intervention, with the overall goal of involving humans at higher levels of analysis and decision making, these services provide additional services that support infrastructure that reports events to operators, supplementing the standard channel reporting functions. They also provide a number of functions for the applications, for example, to assist error handling in the event of problems encountered in an interaction. The architecture follows a tiered model:[1] • The top layer, access channels, comprises workstations and other channels. The next layer is the process layer or integration engine, which defines the various processes executed, message transportation, checking message integrity, perform authentication and identify the needed (external or internal) services to perform the action. • The external services defined in the process layer are executed by internal services in the next layer, which is called the service layer. An external service may be executed by one or more internal service components, and an internal service may be invoked by one or more external services. The internal services are not aware of the characteristics of whatever is calling them; the simply provide a function or functions to the requester. Redeveloping the applications would therefore be a less risky venture than rewriting from scratch. This architecture, driven in large part by business needs, not technology for its own sake.

4. PROPOSED ARCHITECTURE

The proposed architecture is a semantic approach towards elastic service framework for modeling EAI. Semantic services are the main services that address the semantic problem. They are divided into ontology services and mediation services. The principle of ODSOA is based on the semantic services that perform mediation processes in order to resolve mainly the semantic heterogeneity problem. These processes exploit some formal ontologies which take important part in our architecture. Ontology services aims to define the semantic description of services using ontology concepts. In the context of information integration, two fundamental approaches can be selected to achieve semantic integration: shared ontology approach and non-shared (local) ontology approach. In general, we use an hybrid approach that combines the two approaches and that consists to map local ontologies onto a shared ontology. For our purpose, we have chosen a variant of the hybrid approach. This can be motivated by the fact that in a large enterprise of autonomous EISs, the definition of a single set of ontology standards that are suitable for everyone is nearly impossible. Indeed, this can also be motivated by the difficult leveraging when adopting multiple independent ontologies. In our approach, we have defined three major types of ontologies information or data-based ontologies, behavior or functional based ontologies and process or business-based ontologies (figure 1). These ontologies are used in service descriptions such as DSD (Data

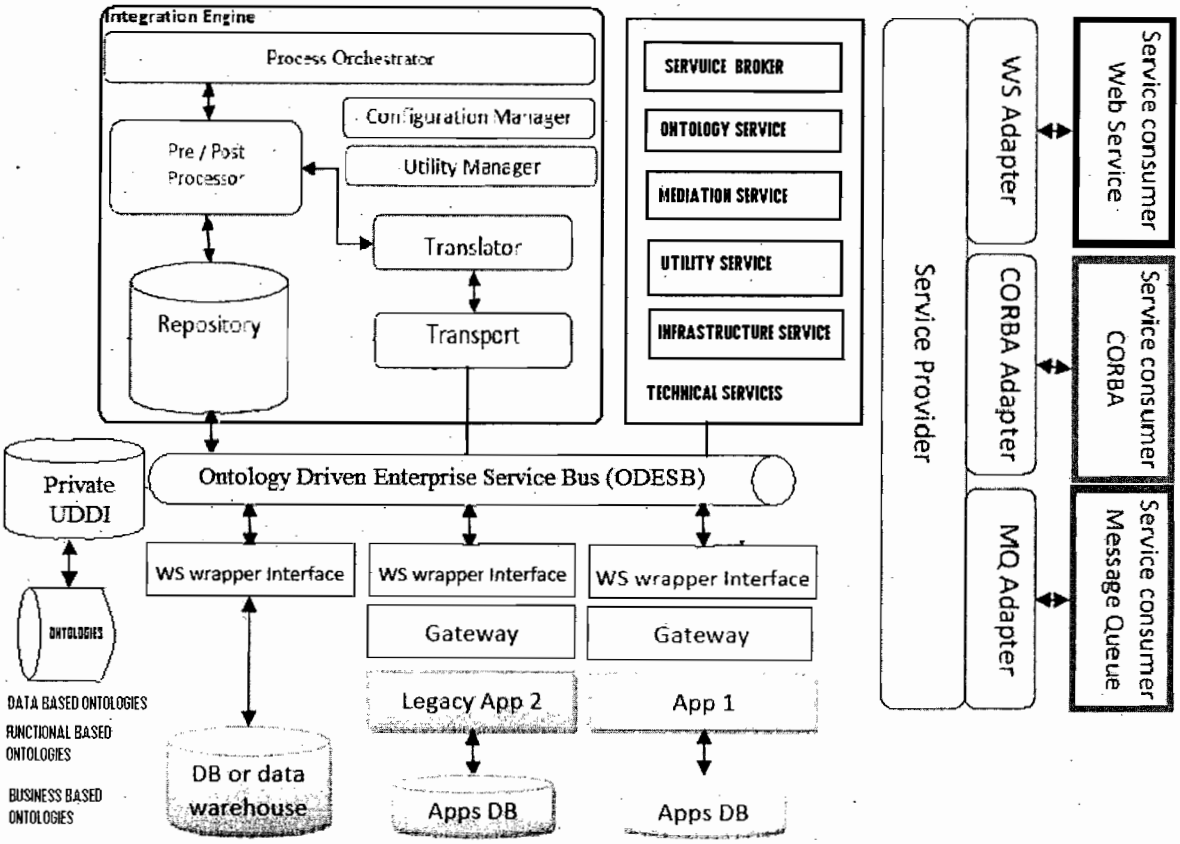

Fig. 6. Semantic driven Elastic service based architecture for EAI

Description), FSD (Function Service Description) and BSD (Business Service Description) to explicit the semantics of respectively DSs, FSs and BSs. In order to semantically describe fundamental services, our approach is based on service ontology, which is an extension of the OWL-S ontology standard (referred to it as OWL-S+). The choice of OWL-S is motivated by the fact that it constitutes a standard and powerful semantic orientation to the description of fundamental services, it completes WSDL (Web Service Description Language) which provides only a description based on a syntactic orientation[9]. In addition to this, we must mention that OWLS has another advantage, which is its compliance with OWL that is chosen as the ontology language for developing our specific ontologies. Our approach mainly focuses on service interface description, and precisely on service profile (OWL-S layer). This latter constitutes the most relevant part of our service ontology, which provides descriptions for fundamental services . Another important extension is the description of *Brokering-services* and blocks shown in figure 6.

Mediation services are generally invoked by service brokers in order to perform matching or resolution of semantic heterogeneity between services. They exploit the descriptions provided by the ontology services described above. Since we adopt an hybrid ontology approach, this require the integration and mediation of ontologies which are performed by Ontology Mediation Services (OMS) and that are based on ontology mapping . This latter is the process whereby two or more ontologies are semantically related at conceptual level. According to the semantic relations[10] defined in the mappings, source ontology instances can then be transformed (or matched with) into target ones. These mediation services aims to mediate respectively between DSs, FSs, BSs and they are based on OMS that match and mediate between different ontologies. To be performed, OMS can exploit two particular utility services that are inference service and matching service. These particular services can be respectively supported by academic or commercial inference engine and matching tool.

5. ADVANTAGES OF PROPOSED ARCHITECTURE FOR EAI

The proposed framework is based on services and internal services are mainly based on web services, it derives all the inherent strengths of service oriented architecture. These advantages have tremendous impact on corporate productivity. The following are the important advantages: • Single point control co-ordination and customization through the "Integration Engine", providing for human intervention, when necessary • Easy to query the business processes and modify them to suit changes in the business environment like a new step in the workflow or an overhaul of the existing workflow. The framework

also facilitates the consolidation of similar applications after a corporate merger or acquisition. All this is made possible because of the use of service-based schemas [7] to model the business processes. • Provides meaningful, interpretable message format for easy documentation. • A unified approach for enterprise application integration that exploit both ontology mediation and web services[8].

6. CONCLUSION

The semantic integration of enterprise information systems is a crucial problem that can concern data, applications and processes. This problem needs semantic mediation and is best resolved in the context of service oriented architectures and elasticity of services through multiple channels. This approach called ODSOI (Ontology-Driven Service-Oriented Integration) aims to extend the current web services stack technology by a semantic layer offering elastic semantic services that can define the service semantics and also perform semantic mediation in the context of EAI.

7. REFERENCES

[1] O.R.Bagheri, R.Nasiri, M.H.Peyravi, P.Khosraviyan Dehkordi, "Toward an elastic service based framework for Enterprise Application Integration ", Fifth International Conference on Software Engineering Research, Management and Applications, DOI 10.1109/SERA.2007.134

[2] Said Izza, Lucien Vincent, Patrick Burlat , "Ontology Urbanization for Semantic Integration: Dealing with Semantics within Large and Dynamic Enterprises" Proceedings of the 2005 Ninth IEEE International EDOC Enterprise Computing Conference (EDOC'05) 0-7695-2441-9/05 $20.00 © 2005 IEEE

[3] Said Izza, Lucien Vincent, Patrick Burlat, "A Unified Framework for Enterprise Integration :An Ontology-Driven Service-Oriented Approach"

[4] Exposing Semantic Web Service principles in SOA to solve EAI scenarios

[5] Francisca Losavio, Dinarle Ortega, Maria Pérez: "Comparison of EAI Frameworks", in Journal of Object Technology, vol. 4, no. 4, May-June 2005, pp. 93-114

[6] Adra Al Mosawi, Liping Zhao and Linda Macaulay, "A Model Driven Architecture for Enterprise Application Integration ", Proceedings of the 39th Hawaii International Conference on System Sciences - 2006

[7] Francisca Losavio, Dinarle Ortega, María Pérez, "Modeling EAI", Proceedings of the XXII International Conference of the Chilean Computer Science Society (SCCC'02) 1522-4902/02 $17.00 © 2002 IEEE

[8] Ashok Kumar Harikumar, Roger Lee, Chia-Chu Chiang, Hae-Sool Yang, "An Event Driven Architecture for Application Integration using Web Services ", 0-7803-9093-8/05/$20.00 ©2005 IEEE.

[9] Xuesong Chen1, Hua Xu, "One Service-oriented Architecture Based Enterprise Application Integration Platform", Feb. 12-14, 2007 ICACT2007

[10] Peter Martinekl), Balazs Tothfalussy2), and Bela Szikoral), "Semantically described services in the Enterprise Application Integration", 30th ISSE 2007,. Budapest, Hungary, 1-4244-1218-8/07/$25.00 ©2007 IEEE