Three Dimensional Database: Artificial Intelligence to eCommerce Web service Agents

R.Vadivel *vadivel.rangasamy@gmail.com*
Department of Computer Science
Karpagam University
Coimbatore, 641024, INDIA

Dr K. Baskaran *baski_101@yahoo.com*
Associate Professor
Department of Computer Science
Government College of Technology
Coimbatore, 641 015, INDIA

Abstract

A main objective of this paper is using artificial intelligence technique to web service agents and increase the efficiency of the agent communications. In recent years, web services have played a major role in computer applications. Web services are essential, as the design model of applications are dedicated to electronic businesses. This model aims to become one of the major formalisms for the design of distributed and cooperative applications in an open environment (the Internet). Current commercial and research-based efforts are reviewed and positioned within these two fields. A web service as a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-process able format (specifically Web Services Description Language WSDL). Other systems interact with the web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards. Particular attention is given to the application of AI techniques to the important issue of WS composition. Within the range of AI technologies considered, we focus on the work of the Semantic Web and Agent-based communities to provide web services with semantic descriptions and intelligent behavior and reasoning capabilities. Re-composition of web services is also considered and a number of adaptive agent approaches are introduced and implemented in publication domain with three dimensional databases and one of the areas of work is eCommerce.

Keywords: Web Services, Semantic Web, eCommerce, Artificial Intelligence, Publication Domain, Dynamic Web, Three Dimensional Database.

1. INTRODUCTION

Currently, Web services give place to active research and this is due both to industrial and theoretical factors. On one hand, Web services are essential as the design model of applications dedicated to the electronic business. On the other hand, this model aims to become one of the major formalisms for the design of distributed and cooperative applications in an open environment (the Internet). Research in the field of semantic web / web service (WS) and artificial intelligence (AI) communities are coming together to develop solutions that will take us to the next and more mature generation of the web application. The composition of web services to create a value-chain greater than the sum of the parts is a key part of what can be expected. The fulfillment of the vision of the web as an information-providing and world-altering provider of services is not far away. More futuristic is the notion of serendipitous. In both visions the services and outcomes may be the same.

However, the difference between the two visions is that the first can be achieved through static and manual solutions and the second requires dynamic and automated solutions. While helpful

for the first, the addition of semantic content on the web is essential to enable automatic discovery and composition of multiple services. It is natural that earlier work in the field of AI will assist in realization of the (artificially) intelligent web. The work on the Web Services Modeling Framework (WSMF) is an example of AI being applied to this field. WSMF offers the combined use of ontology, goal (problem-type) repositories, web service descriptions and mediators to handle interoperability issues. The agent community, which is primarily AI-based, has also been actively conducting WS related research.

Our own distributed agent-based work and the Agent Factory, originates from our earlier AI research into complex knowledge based systems and generic task based configuration. On the one hand, our work on planning and automated configuration offers a way of composing eCommerce web services. On the other hand, WSs potentially provide us with components needed to achieve an implementation of our design. Through the addition of techniques from the Semantic Web community, the benefits of combining our agent technology with WSs has been mutual.

This paper offers a review of research that overlaps the fields of WS and AI. In the following section we describe web services and the need for semantics to be added. In section B we look at how the Semantic Web communities, within the field of AI, are offering semantics. In section C we present AI-based research to address the discovery of WSs. In section D we consider both commercial and AI based techniques for WS composition. In section E, the notion of re-composition of WS is considered and how adaptive agent technology, including our own, can address this problem. We conclude with future directions for the role of AI in the web services field.

2. RELATED WORKS

Implement artificial intelligence concepts to agent's communication and improve the efficiency of the communication between the agents. We are going implement this concept to publication domain and that domain application has used three dimensional databases architecture has been played major rule on the creation dynamic fields. Database architecture for creating dynamic web controls is a three dimensional structure, where we use three terms static, meta and dynamic. Here Static data is generally creating the tables and fields to the database. Meta data is a bridge between static and dynamic data. Dynamic data is the dynamic resultant tables or views that the user needs. An output of database is XML format and it contains data definition and data values. XSL is a presentation part which transforms XML data to output HTML.

In three dimensional databases has used two types of SQL statements Static and Dynamic. Static SQL is SQL statements in an application that do not change at runtime and, therefore, can be hard-coded into the application. Dynamic SQL is SQL statements that are constructed at runtime; for example, the application may allow users to enter their own queries. Thus, the SQL statements cannot be hard-coded into the application.

2.1 Web Services

Web services are typically application programming interfaces (API) or web APIs that can be accessed over a network, such as the Internet, and executed on a remote system hosting the requested services.

Web services are a new way of connecting business. Web services are platform-neutral and vendor-independent protocols that enable any form of distributed processing to be performed using XML and Web-based technologies.

2.1.1 Just-in-time Integration

The Web Services architecture describes the principles behind the next generation of e-business architectures, presenting a logical evolution from object-oriented systems to systems of services. Web Services systems promote significant decoupling and dynamic binding of components: All components in a system are services, in that they encapsulate behavior and publish a messaging

API to other collaborating components on the network. Services are marshaled by applications using service discovery for dynamic binding of collaborations. Web Services reflect a new service-oriented architectural approach, based on the notion of building applications by discovering and orchestrating network-available services, or just-in-time integration of applications.

2.2 Semantic Description of Web Services

WSDL, SOAP and UDDI are seen as steps in the right direction but ones that will fail to achieve the goals of improved automation and interoperability, because they rely on a priori standardizations and require humans in the loop. To support automated reasoning, knowledge representations (such as markup languages) will be needed that express both data and rules for reasoning. The ability to dynamically locate and compose web services based on their semantic description will rely on the richness of the description and the robustness of the matching techniques used. Ontology will be used to enable definition and comprehension of meaningful content. These are the concerns of the Semantic Web community. Additionally, agents will be needed to interpret the content and transform user requests into optimized delivered solutions. The Intelligent Brokering Service for Knowledge-Component Reuse on the WWW (IBROW)4 can be seen as a forerunner of the Semantic Web. In IBROW problem solving methods (PSMs) and ontologies were the components being configured, the current focus is on WS configuration. PSMs and ontologies when used together are also capable of delivering services. The most significant work that has been done to describe web services has been conducted by the DAML-S coalition. The matching of service providers and service requesters via semantic descriptions of the services are key goals of this work. DAML-S uses the DAML+OIL specification language (which extends the weak semantics of RDF(S)) to define a number of ontologies that can be specifically used to describe web services. DAML-S is built on the AI-based action metaphor where each service is either an atomic/primitive or composite/complex action. Knowledge preconditions and knowledge effects are handled via the inputs and outputs of the web service. The DAML-S coalitions are providing solutions to work with current WS standards. For example, a DAML-S service grounding definition can be mapped to a WSDL definition of the service. A number of approaches to service discovery and composition that we discuss in the following sections use or extend the DAML-S web service ontology.

2.3 Discovering Web Services

Discovery involves locating and/or matchmaking against some selection criteria. An earlier AI system, Lark, which involved annotation of agent capabilities to enable them to be located and brokered, clearly solved a problem similar to the discovery of WS by a middle agent. This work has developed into the DAML-S Matchmaker5. To support matchmaking a number of filters may be configured by the user to achieve the desired trade off between performance and matching quality. These filters include: word frequency comparison, ontology similarity matching, ontology subsumption matching, and constraint matching.

Offer an alternative to sequential searching when matchmaking an agent with a service request. They point out that finding possible partners via matching of service advertisements with requests is not enough. To support runtime interactions we need smarter behavior to handle components that are not quite what was requested and combining several partial components to meet the original request. The solution to overcome sequential searching is the conversion of the concepts into number intervals and the use inheritance hierarchies to determine subclass and equality relations. A generalized search tree is used to handle partial matches.

The feasibility of matchmaking largely depends on the annotation of web services. AI can also be applied to this problem. A number of markup tools have been developed for document markup and these could be applied to the semantic description of WSs. The SHOE Knowledge Annotator [19] uses ontologies to guide knowledge annotation. To produce RDF-based markup, COHSE or AeroDAML can be used. These approaches start with descriptions in DAML+OIL and DAML, respectively. These approaches support automatic conversion of markup languages but do not support information extraction or automatic mark-up. OntoMat does support some form of

automated extraction of semantics. OntoMat combines the resource with its DAML-S markup. The MnM approach additionally stores the annotations in a knowledge base. Automated markup in MnM is achieved using techniques from knowledge engineering; machine learning and natural language processing have developed a query language that is used to find services.
The solution to finding services is to first describe the service using the process ontology with the assistance of the MIT Process Handbook. The Handbook is large and allows reuse to assist in ontology definition. Next, the ontology is indexed by breaking it down into its components such as attributes, ports, dependencies, subtasks and exceptions. The requester can form a query in the query language that will use the index to find matches.

Clearly AI is already contributing solutions for locating, matchmaking, querying and annotation of WS to facilitate their discovery. Discovery of web services is an important issue as it is a prerequisite to their use. However, the real value of web services lies in their composition.

2.4 Composing Web Services

Web service composition can be simply defined as: “the problem of composing autonomous services to achieve new functionality”. WS composition is not just an alternative to application development but a means of reducing the application backlog problem because: many services are moving online; integration is easier since WSs conform to the HTTP protocol and many independent providers have related services that need to be combined to satisfy user requirements. The rigidity and lack of intelligence of current solutions has spawned a number of research projects from a number of other research fields.

The work by has arisen from experience in the distributed systems and networking fields. They have developed the Infrastructure for Composability at Runtime of Internet Services (ICARIS). They have extended WSDL to develop the Web Services Offerings Language (WSOL). They offer flexibility and adaptability but their approach is very alternative. Instead of trying to solve the problem of how to find services dynamically and combine them, they focus on the situation where providers and requestors are already matched but will at times either make changes to their services or requests. A service is seen to have numerous offerings. The functionality will be the same but the constraints will differ such as authorization rights and QoS. They suggest that a limited number of classes of services be offered and described. Then using WSOL they are able to specialize the classes into offerings. Their solution offers dynamic switching between offerings. From a commercial point of view the notion of offerings makes sense as customers probably prefer to do business with companies they already know and businesses want to maintain their existing client base.

The work at Hewlett Packard laboratories on eFlow is similar in that dynamic composition involves automatic adaptation of the configuration at runtime according to the requests of the individual customer. The approach is driven by the view that composition adds value but to stay competitive, composition needs to be dynamic as services offered need to adapt to stay competitive. Their goal is to allow dynamic change in service processes with no or minimal human intervention. While they take a business process perspective they point out that web services are less static, predictable or repetitive compared to “traditional” business processes. Similar to most current commercial solutions, dynamic composition is made possible due to the use of a central repository that has clients and providers already attached to it.

The notion of generic solutions that are customized according to user constraints is a recurring theme in much of the literature. Also look at composition as the selection of possible services based on user specified criteria. They offer a centralized, pipes and filters architecture with two main components: a composer (user interface) and an inference engine (IE) component (which includes a knowledge base of known services). The inference engine is an OWL reasoned and includes axioms to find all relevant entailments, such as the inheritance relation between two classes which may not have been made explicit. The user identifies some criteria that the service must satisfy. The matchmaker (IE) selects services that might be suitable based on those criteria and the composer shows them to the user. Suitable services for composition are ones whose

output can be an input to a selected service. While execution of WS may be performed automatically, the actual task of composition is performed by a human using the services suggested by the system.

Model-based reasoning is a common technique employed in AI approaches. In SWORD entity relationship modeling of services is performed by "base service modelers" to produce a "world model". After building a world model for each service, a composition model is developed that models each service as an action. An expert system is used to automatically determine if the composite service can be created with existing services and if so a plan of execution is generated.

In summary, a number of solutions are offered to provide web service composition. The approaches described in this section show that composition can be assisted through the use of class definitions, inheritance hierarchies and model and rule-based reasoning. In many cases, decision making is left to humans. The only automated composition offered is in limited situations where a central repository is used and the requestor and provider are part of the same system. However, the web is distributed in nature. Intelligent reasoning and collaboration between services is needed to handle this complexity. Agents are capable of both.

2.5 Agents and Web Services

The autonomous and reasoning capabilities of agents make them well suited for handling cross-organizational decision making. For example, agents can be used to (re)negotiate contracts which would then require: determination of which processes are needed to fulfil the contract; creation of new business processes; and adaptation of existing business processes. Two main agent-oriented approaches exist: use wrappers to make WS behave like agents and; using agents to orchestrate WS.

2.5.1 Adding Behavior to WS Via Agents Wrappers

WS are componential, independent, software applications similar to agents. However, agents are also reactive, social and capable of reasoning. If we wish web services to work together, we need to give them social and reasoning capabilities. This can be achieved by wrapping a service in an agent. In the work of, a composition language is used to create an agent wrapper which allows services to collaborate. The created agent has first–order reasoning abilities that have been derived from the DAML-S description of the service. This then allows one agent wrapped service to know what other agent-wrapped services are capable of doing and whether they can assist in the service/agent meeting its goals. Also offer an agent-based wrapper approach to web services. They have developed a tool for creating wrappers so that web sources can be queried in a similar manner to databases. They then use an interactive, hierarchical constraint propagation system to perform integration. As in, the end user interacts via a GUI to manage the orchestration. The Racing project6 offers a mediator architecture also using agent wrappers that are structured into a hierarchy. A number of different agent wrappers are supported: user, query translation, query planning, resource wrapper, ontology, matchmaking, and cloning and coordination agents. The use of agent wrappers is a way of allowing multi-agent system technology to be applied to web services.

2.5.2 Composing Web Services Using Agents

The work of combines ideas from the Semantic Web, Knowledge Representation and Agent communities to allow WSs to be composed. Their goal is to "construct reusable, high-level generic procedures, and to archive them in shareable (DAML-S) generic-procedures ontologies so that multiple users can access them". In the approach, WSs and user constraints are marked up in DAML-S. A generic task procedure is selected by the user and given to the DAML(-S) enabled agent, who customizes the procedure according to the user specific constraints. The generic procedures are written in an extended version of ConGolog, a situation calculus agent programming language, and executed using a Prolog inference engine. Others provide agent-oriented languages for web service description. Propose an Agent Service Description Language (ASDL) and Agent Service Composition Language (ASCL). ASDL is an extension to WSDL and

captures external behavior via a finite state machine. Their work is based on the argument that composition requires more than description of the data, but also requires a strong representation of actions and processes. A number of approaches are focused on the design of agent systems with web services as the components have developed WARP (Workflow Automation through Agent-based Reflective Processes) that uses the XML and WSDL standards. The goal is automatic configuration and management of low-level services (components). The software engineering development process that has been developed is semi-automatic involving multiple software agents and a human workflow designer. They support visualization of the process based on activity diagrams in UML.

2.5.3 (Re-)composition and Adaptable Agents

The ability of agents to adapt according to changes in system requirements and the environment is important to enable dynamic and reactive behavior.
Agents may be adapted in a number of different ways. The knowledge and facts that an agent uses may be adapted for example the agent may use a client profile that changes according to the clients activities (e.g. this type of adaptation typically involves machine learning, e.g. An agent may also adapt its interface according to the platform on which it is being used (e.g.[brand]. A third type of adaptation, and the type of adaptation we are concerned with, is adaptation of the agent's functionality. There is limited work in this area. Semi-automatic agent creation tools such as AGENTBUILDER, D'AGENTS/ AGENT/TCL, ZEUS and PARADE could possibly be extended to support agent adaptation.

Following the use of compositionality in the major software engineering paradigms (e.g. functional programming, object-oriented programming, component-based programming and the Factory design pattern, we have developed an Agent Factory. The approach is based on the use of components, the general agent model (GAM) and the DESIRE formal knowledge level modeling and specification framework for multi-agent systems. Our agent (re-)structuring approach allows an agent to automatically adapt by reusing existing components. Our approach is a combination of process-oriented and object-oriented approaches by treating processes as the 'active' parts of our agent, which are our agent components and classes as the 'passive' part of our agent, which are the data types used in the agent components. We are currently exploring whether DAML-S descriptions of web services are adequate for automated configuration of web services by the Agent Factory.

the Agent Factory and based on the notion of design patterns, assists human designers in functional design, and the configuration of software components to fulfill the conceptual design specified by the designers, depending on the agent platform that is to be used. Our approach does more: it automates the creation and redesign of both the conceptual and operational design based on the requirements on function, behavior and state of an agent. Our use of web services as components is a further distinguishing feature.

While not currently working in the WS area, the Adapt agent approach, bring together adaptive workflow and agent research. They consider how agents can be used to collaborate to perform a workflow and make workflow more intelligent and how workflow can be used to organize a set of agents and coordinate interaction between people and agents.

The reuse of knowledge has also been a widely researched topic and the creation of libraries of problem solving methods and generic task models offer a similar idea to the functional components in our agent factory. The IBROW project, mentioned earlier, has even more in common with our approach by semi automatically configuring intelligent problem solvers using problem solving methods as building blocks. They use mappings to act as glue between the components which are modeled as CORBA objects. Unlike our approach, their architecture is restricted to specific languages and architectures, they only support semi-automation and they do not distinguish between conceptual and implementation level designs.

3. EXPERIMENTAL RESULTS

In this section we evaluate the performance of the proposed artificial intelligence to eCommerce web service agent. We have created common web service application to integrate the all web and windows application who want to integrate eCommerce application to their application. Refer Fig 1 – 3 work flow of AI based eCommerce web service. Solis architecture has used three dimensional database that is Static, Meta and Dynamic. Here Static data is generally creating the tables and fields to the database. Meta data is a bridge between static and dynamic data. Dynamic data is the dynamic resultant tables or views that the user needs. It's provided three main areas of functionality – self-updating interface on the web, robust database administration, searchable front-end for end users. That system is designed so that dynamic data at the core of the integrated system is available in any output or view. The data administrator has control over the data content, various templates and user permissions, thereby giving an unrivalled level of flexibility and control in content collection, management and presentation.

3.1 Application Overview

The Order Management System (eCommerce) has been written to provide a common means to create simple orders and process credit card transactions. The first version works only with PayPal's PayFlowProw service but can be updated to work with other online merchant services (e.g. Authorize). When the need to use an alternative provide comes up we'll code the core library accordingly. Any changes here will not affect the way you use the service.

3.3.1 Not a User Management System

The system doesn't offer any user management capabilities like sign in. It assumes the calling application knows who the user is let's it take care of any user authentication required. When you're coding your shopping carts you need to handle all of this. The Order Management System simply provides the relevant methods to create Processing Sessions, Shopping Carts etc you just need to implement them.

The Order Management System does keep track of "users" (customers) through the use of an email address. This is the primary means of identification and is required before you can process any payments.

3.3.2 Typical Lifecycle / Process Flow

It's important to understand how the Order Management System (OMS) works so you can make use of the methods in the most efficient way.

The first thing you'll need to do is to create a Processing Session in the OMS. This is done by calling the GetSessionForKnownUser() method and passing in the email address of the current user. The OMS will create a session and a shopping cart for the user. It's recommended that you store the ID of the session in your application cookie or in your database so it can be reused. It's not efficient to create a session every time!

To retrieve the shopping cart you simply call the GetShoppingCart() method. To add an item to the shopping cart simply call the AddItemToShoppingCart() method passing in a properly constructed ShoppingCartItem object. To remove an item from the shopping cart simply call the RemoveItemFromShoppingCart() method passing in the item to remove. You can also empty the shopping cart by calling the EmptyShoppingCart() method.

3.3.3 Ready to Checkout

Once you've populated the shopping cart, authenticated your user you're ready to process the transaction and turn the Shopping Cart into an Order. To do this you must create a PaymentProcessingKey. Think of this is a temporary key allowing you to make a credit card transaction. To create one you call the GeneratePaymentProcessingKet() method passing in the session. It will configure it with the session and the associated shopping cart ready for processing.

3.3.4 Payment Information Page

This page is the one responsible for taking the credit card information and processing the payment through the online payment gateway (e.g. PayFlowPro). The Order summary is displayed at the top of the page so the user can make sure they're purchasing the correct item(s). The next section prompts for the credit card information including the CVV2 security code location on the credit card.

The last section prompts the user for the billing address that's associated with the credit card they're using. If the user is purchasing physical goods they should also populate a shipping address. If the order consists of only electronic items the shipping address can be left blank.

3.3.5 Processing the Payment

Once the user is happy that all the information has been entered correctly they should click the "Purchase Now" button to initiate the payment transaction. Processing payments is actually using a two step process:

1. The first step is to authorize the payment. The reason we do this is to basically test to see if the payment information is correct and that the payment card will accept the new payment being attempted without actually taking the funds. The reason we do this is to make sure the transaction will succeed. If this step fails we send the user back to the payment information page and display them the error. It basically means that we'll never process an order unless the payment succeeds.

2. The second step is to then retrieve the actual funds allocated during the first authorization step. At this point we're 99.9% confident that the transaction will succeed because the authorization was successful.

After a successful transaction the system performs some cleanup routines and processes the order:

1. Updates the status of the order to Complete
2. Constructs an invoice and sends this to the user
3. Constructs a notification email and sends this to the person setup in the installation configuration
4. Calls the Call-back page defined int the installation configuration. This page is located on calling application and is generally responsible for firing any triggers based on the products that were just purchased. For example it might need to perform an upgrade of a profile or add a new feature.

Once all this is complete the user is sent to the Order Confirmation Page where a summary of the order is presented.

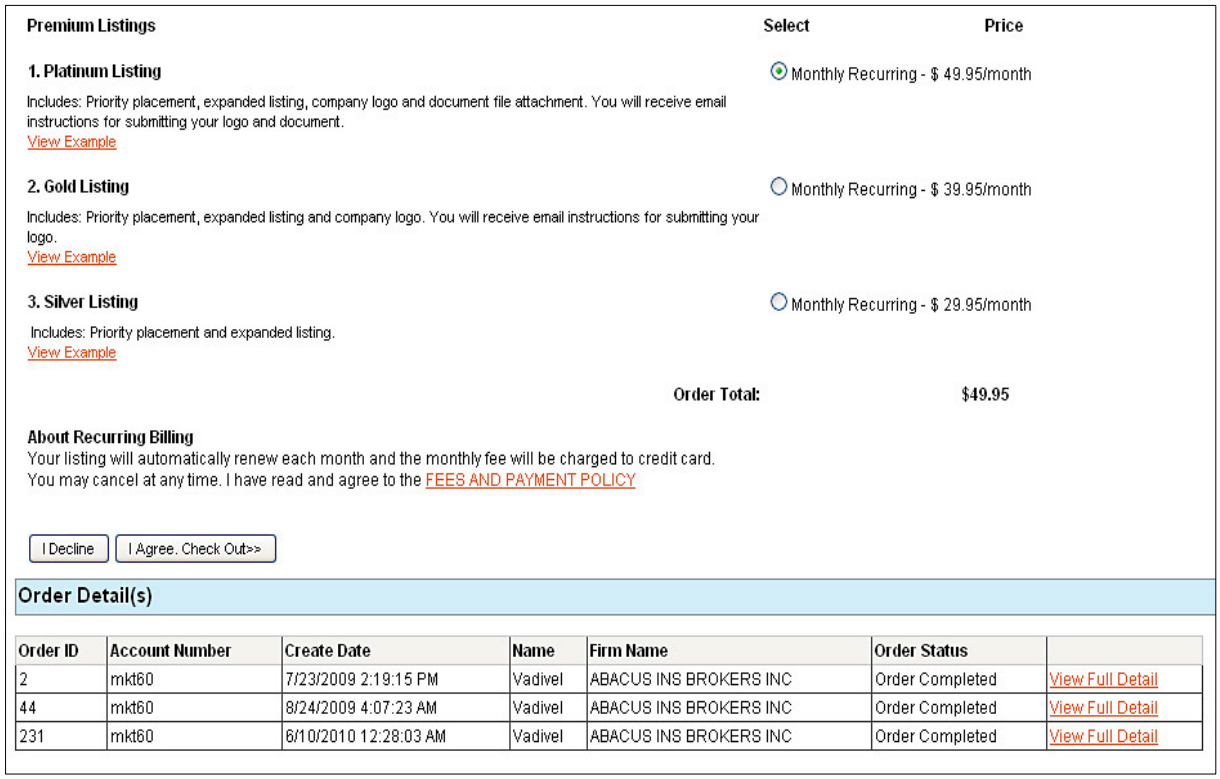

FIGURE 1: Product select and Checkout page

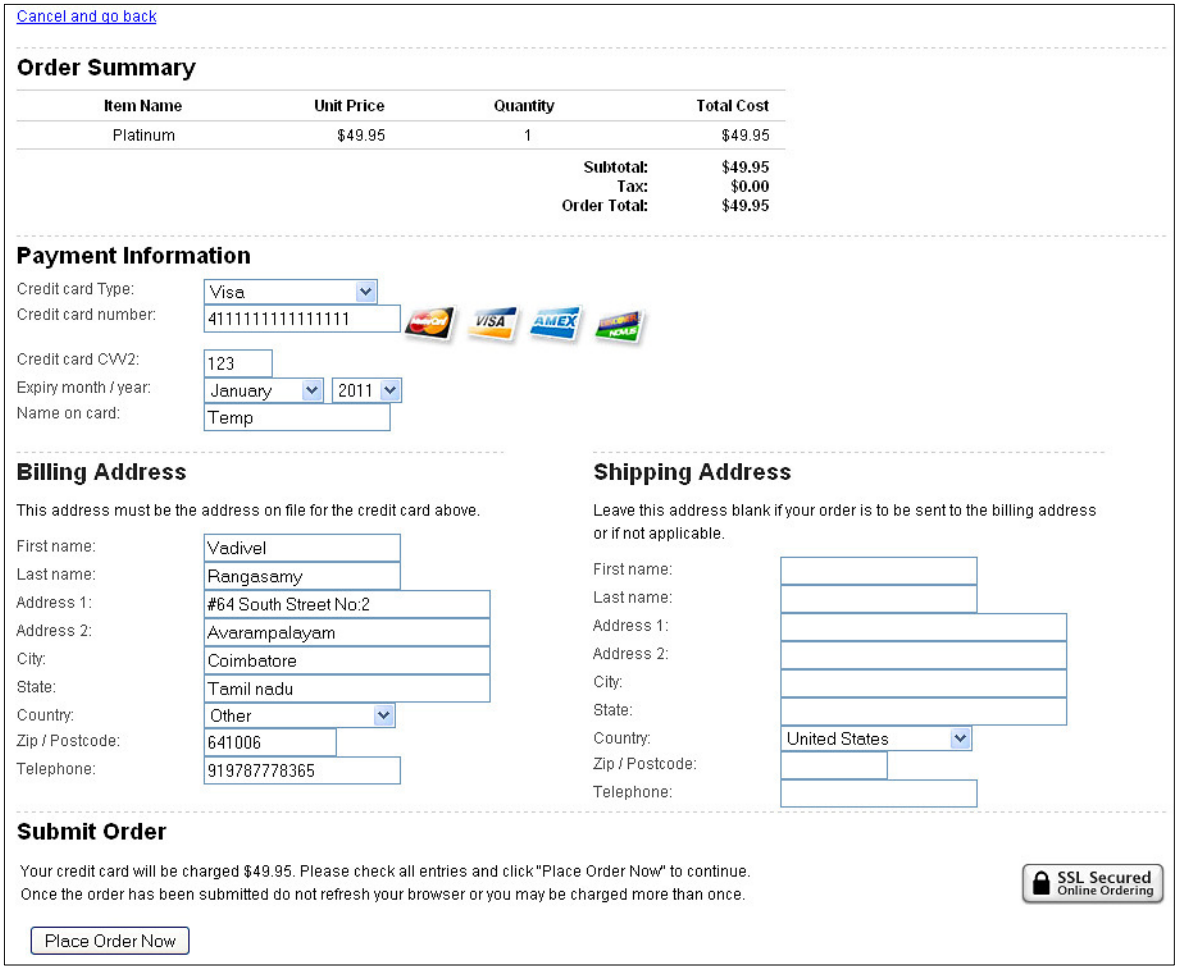

FIGURE 2: Credit card and Billing address page

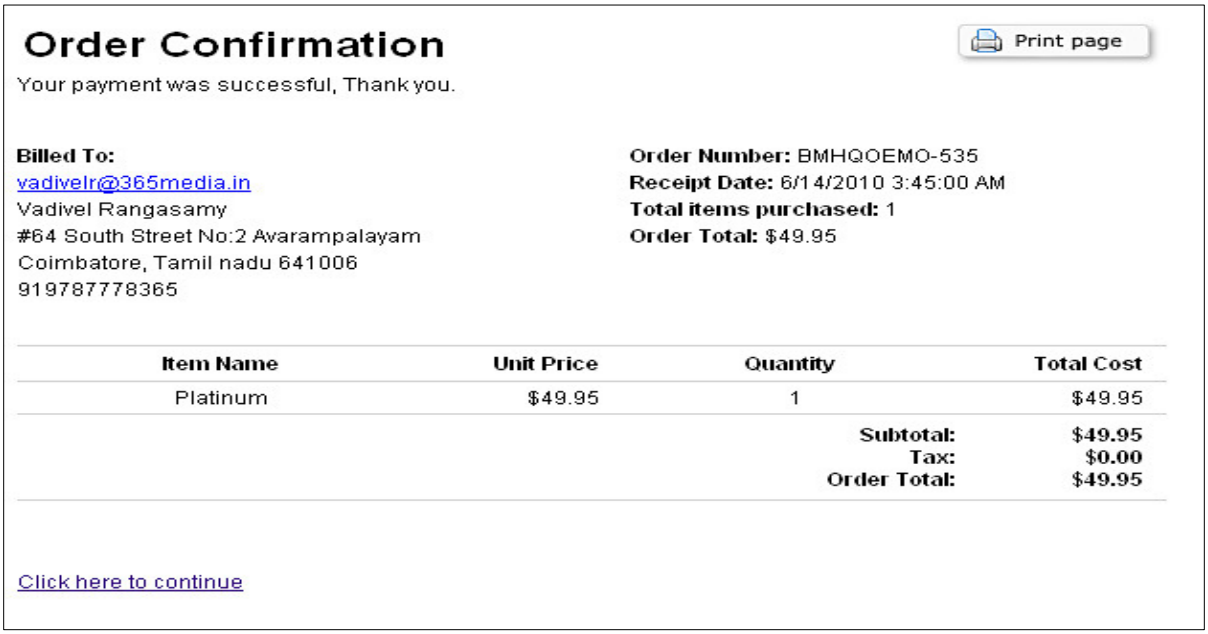

FIGURE 3: Order confirmation page

4. CONSLUSION & FUTURE WORK

The work of the Semantic Web community to provide semantic description of web services will play a key role in enabling agents to automatically compose web services. In this eCommerce application has implemented in embedded windows and web applications with cross-platforms and it's successfully interoperability of applications. A standard communication between the agents is clearly defined and very less amount of data loss.

Existing agent platforms may need to be adapted to handle the specific requirements of web services. But in this system with no trouble to adaptable all kind of computer applications and tested in real world applications. The RETSINA functional architecture includes four basic types of agents: interface, task, information and middle agents who communicate via a special agent communication language. Each of these agents includes four reusable modules: communication and coordination, planning, scheduling and monitoring. The middle agent plays a critical role in matching providers with requesters and is offered as a solution to the heterogeneous nature of agents over the web.

In future work will continue on artificial intelligence to natural language technology research will assist discovery of web services and agents will play an important role in using web services to satisfy user requests.

5. REFERENCES

[1] Boutrous Saab, C.; Coulibaly, D.; Haddad, S.; Melliti, T.; Moreaux, P.; Rampacek, S. "An Integrated Framework for Web Services Orchestration", Idea Group Publishing, 2009

[2] Raymond Y. K. Lau, "Towards a web services and intelligent agents-based negotiation system for B2B eCommerce", Elsevier Science Publishers B. V.,October 2007

[3] Sabou, M., Richards, D. and van splunter, S. An experience report on using DAML-S , Workshop on E-Services and the Semantic Web, Budapest, Hungary, May, 2003

[4] B.Y. Wu and K.M. Chao. Spanning Trees and Optimization Problems. CRC Press, New York, USA, 2009.

[5] Xia Yang Zhang Qiang Xu Zhao Zhang Ling, "Research on Distributed E-Commerce System Architecture", IEEE ,August 2007

[6] Lixiao Geng Zhenxiang Zeng Yajing Jiang, "Research on E-Commerce personalized service based on intelligent agent technology", IEEE, November 2008

[7] T. Finin, J. Mayeld, C. Fink, A. Joshi, and R. S. Cost. Information retrieval and the semantic web,January 2004.

[8] Scott Short, "Building XML Web Services for the Microsoft .NET Platform", Microsoft Press, 2002

[9] James Murty, "Programming Amazon Web Services", O'Reilly Media, March 2008

[10] http://www.daml.org/ontologies/, daml ontology library, by daml.

[11] http://www.schemaweb.info/, schema web.

[12] http://www.semwebcentral.org/, semwebcentral, by infoether and bbn.

[13] http://www.w3.org/2004/ontaria/, ontaria, by w3c.