Reconfigurable Web Service Integration in the Extended Logistics Enterprise

Alex Talevski, *Member, IEEE*, Elizabeth Chang, *Member, IEEE*, and Tharam S. Dillon, *Fellow, IEEE*

***Abstract*—Transportation and warehousing logistics are activities that require strong information systems and computer support. This requirement has grown with the advent of e-commerce. Companies such as FedEx and UPS now allow their customers to track and monitor the fulfillment of their requested services on the Internet. In order for such a system to be effective, the goods need to be handled by the one corporation, with an integrated system. This is rare in the case of Small to Medium sized Enterprise (SME). With the advent of Business to Business (B2B), and Partner to Partner (P2P) e-commerce, there has been an increasing tendency for SMEs to set up consortia that represent several players in a given field in order to contend with larger competitors. This paper deals with the concept of an extended logistics enterprise and explores the software engineering issues underlying the development of such complex systems.**

***Index Terms*—Extended logistics enterprise, service integration, service reconfiguration, systems interoperability.**

I. Introduction

THE TERM "enterprise application" defines a class of applications that automate key business functions, such as managing supply chains and customer relationships. Custom enterprise applications constitute the majority of software systems and development. The Enterprise Application Market (EAM) has been predicted to grow to US$78 Billion in 2004 where the E-Business Relationship Management (ERM) and Supply Chain Management (SCM) segments are of highest economic importance, both representing more than 15% of the market share.

The advent of the web has provided mechanisms for binding organizations together, carrying out sales over great distances at any time through new modes of marketing and enabled partnerships that were previously inconceivable. A consequence of this connectivity and information richness is that we are now faced with an increasingly competitive and rapidly evolving environment. We now require new business paradigms and organization forms that transcend the previous static, closed models and move to flexible, re-configurable, and collaborative models that are able to respond to environment dynamics inherent within the networked business economy.

Manuscript received August 13, 2004.

A. Talevski was with the Department of Computer Science and Computer Engineering, La Trobe University, Bundoora, Vic. 3083, Australia. He is now with the School of Information Science, Curtin University, Perth, WA 684, Australia (e-mail: Alex.Talevski@cbs.curtin.edu.au).

E. Chang is with the School of Information Science, Curtin University, Perth, W.A. 6845, Australia (e-mail: change@cbs.curtin.edu.au).

T. S. Dillon is with the Faculty of Information Technology, University of Technology, Broadway, N.S.W., Australia 2007 (e-mail: tharam5@it.uts.edu.au).

Digital Object Identifier 10.1109/TII.2005.844421

The extended enterprise represents a new business paradigm and organizational form. Several factors characterize an extended enterprise, namely, the following.

- **Collaboration**: An increasingly collaborative approach between individual enterprises.
- **Infrastructure**: A strong information infrastructure that extends beyond the original closed walls of the individual enterprise.
- **Electronic interaction**: Consumer/provider services coupled with high connectivity and electronic handling of information.
- **Reconfigurable and adaptive**: Ability to self organize, and reconfigure the underlying architecture.
- **Multichannel marketing**: Use of multiple channels for sales and marketing.
- **Data mining**: Capture and utilization of business intelligence from data and smart information use; these features are being used by successful organizations, in collaborative supply chains and virtual logistics consortia for marketing, strategic partnerships, and selling of services.

A key factor in the success of such extended enterprises is the creation of the underpinning information technology infrastructure in order to integrate collaborating business partners. Inter-changing information between logistics partners is a key factor in being competitive.

A. Consortia

In recent times in B2B and P2P e-commerce, there has been an increasing tendency to set up consortia among SMEs in order to contend with larger competitors. Consortia consist of organizations in a given field that get together and produce a single site. These consortia fall into two classes.

- Companies getting together to form a single site to increase traffic through this site as compared to other competitor's sites.
- Companies that have their own regions of operation but could benefit by providing a service to customers that extends beyond their region of operation.

A new form of collaboration is likely to develop in the near future leading to the concept of a virtual extended logistics provider. This virtual provider assembles many logistic providers into strategic alliances. Alliance partners combine their facilities to achieve a warehousing and transportation network that is geographically widely distributed, and provides a greater range of services. Thus, an integrated extended logistics hub and portal that facilitates the utilization of these products and services is essential. Such a central portal may

offer warehousing, logistics, auction, virtual market, and other online facilities from a range of integrated partners.

B. Requirements

This new class of virtual logistic providers will require a new level of IT support. Due to the diverse environment that this class of consortia operates in, there exists a special need for runtime inter-organizational system integration, information exchange and communication. Collaborating services need to be identified by each partner and then integrated with the central portal. However, several factors represent major issues in developing such a hub and portal. Both technical and other environmental issues exist.

C. Technology

A problem faced in developing a system such as the one discussed, is the complexity of service integration and reconfiguration occurring on different layers [11]. The heterogeneity in these layers is primarily caused by different:

— operating systems;
— hardware devices;
— network technologies;
— middleware technologies;
— communication and interaction protocols;
— programming languages;
— services and interfaces;
— underlying architectures;
— data and document formats.

Other problems include:

— business processes and procedures;
— business requirements and wants;
— business structures;
— business cultures;
— standards (laws, languages, documents, etc.).

This problem becomes a multidimensional one when nonfunctional properties such as security, availability, and transactions, are interwoven within the application. One of the most challenging aspects of this problem is that collaboration must take place despite the disparate systems being used.

II. Existing Approaches

Existing solutions of partner integration and collaboration have utilized mostly middleware-oriented approaches.

— **Server side redirection**: Server side redirection is used to readdress a user from a native site to a foreign one when a user's requested services cannot be satisfied locally. This type of an approach is typically simple to setup, and has low cost. However server side redirection is a non integrated approach where no collaborative business processes are considered.
— **Remote method/procedure invocation**: Remote method/procedure invocation systems typically employ a client/server infrastructure that increases the interoperability, portability, and flexibility of an application, by allowing the application to be distributed over multiple heterogeneous platforms. However, such systems incur high setup costs, have no service discovery repository and a non standardized interaction scheme is used.
— **Third party message brokers**: Third-party message brokers utilize an open framework that specifies how organizations can exchange business documents in a straightforward way. Interchange standards define business documents such as invoices, bills, and purchase orders. These brokers are normally well-tested solutions that are available off-the-shelf. However, the document formats that are used cannot be easily read by a person and business processes are not considered.
— **Component-based technologies**: Component-based technologies allow for both communication and manipulation of objects over a network connection. They provide a framework for creating, distributing, and managing components. Disparate components communicate through an interface broker. These technologies have matured significantly over the years, but currently do not provide standardized interaction protocols, and are generally incompatible.

The technologies mentioned above are merely tools that facilitate point-to-point connection and communication between organizations. On their own, they do not provide a framework for service integration and reconfiguration. Furthermore, these technologies do not yet provide end-to-end solutions that support extended enterprises.

III. Service-Oriented Architecture (SOA)

A service is a unit of work done by a service provider to achieve desired end results for a service consumer. Services map to distinct business domains. A software service is generally implemented as a course-grained software entity. This service typically interacts with applications and other services through a loosely coupled (often asynchronous), message-based communication model. Interaction with services is performed over well defined interfaces that are considered consumer/provider.

A SOA is a way of connecting applications across a network via a common interaction protocol. SOA is an architectural style whose goal is to achieve loose coupling among interacting software agents by employing two architectural constraints.

— A small set of simple and ubiquitous interfaces to all participating software agents.
— Service interaction is constrained through a structured interaction scheme. A schema limits the vocabulary and structure of messages. This schema allows new versions of services to be introduced without breaking existing services.

SOA defines how those service interfaces are located, executed, managed, monitored and secured. The interfaces are often published in a directory which allows application developers to browse collections of services, select those of interest, and assemble them to create the desired functionality. SOA encourages that larger applications are divided into smaller discrete modules, and that these fine grained modules are used to produce course-grained services that can be easily integrated by others. In this way, services and clients can be changed independently of one another. In theory, this lets developers

map distinct business process as services that can be chained together in order to realize certain collaborative behavior.

A service-oriented architecture requires three fundamental operations: publish, find and bind. Companies implementing Web Services may take on Service Provider, Broker, and/or Requestor roles. This concept of publish-find-bind is shown in Fig. 1.

A service is a course-grained software entity. Interactions with a service are governed by an interface contract. SOAs are comprised of four primary roles.

— **Service provider**: A service provider implements a service specification. It publishes this specification via a service directory, and is utilized by a service consumer. Traditionally, this is referred to as a server.
— **Service consumer**: A service consumer finds provided services which are published by a service directory that it binds to. Traditionally, this is referred to as a client.
— **Services directory**: A service directory is a specific kind of service provider that acts as a registry for service information, and allows for the searching of service provider interfaces and service locations.
— **Service broker**: A service broker is also a specific kind of service provider that can pass on service requests to one or more additional service providers.

In order to efficiently design, develop, and utilize services, a few best practice characteristics should be followed.

— **Coarse-grained**: Operations on services should be implemented to encompass more functionality and operate on larger data sets, compared with component-interface design.
— **Interface-based design**: Services implement separately defined interfaces. The benefit of this is that multiple services can implement a common interface and a service can implement multiple interfaces.
— **Discoverable**: Services need to be discoverable by unique identity, interface and by service type.
— **Single instance**: Unlike component-based systems, which may instantiate multiple components as required, a service is a single, always running instance.
— **Loosely coupled**: Services should interact by using standardised, system independent, message-based methods such as XML technologies.
— **Asynchronous**: Asynchronous message passing is recommended. It is, however, not absolutely required.

A. Objects, Components, and Services

Structured design processes when large scale systems are concerned have in the past revolved around function/data decomposition. However, such systems are susceptible to high maintenance costs as even minor modifications may result in the introduction of bugs throughout the software. Systems built using function/data methods often exhibit characteristics of poor quality. More recently, the object-oriented (OO) approach has been adopted to provide a more natural progression to a solution. By using the OO paradigm, changes to the system can be localized, and therefore will have little side-effect on the overall system. This typically results in a system of higher quality and lower cost.

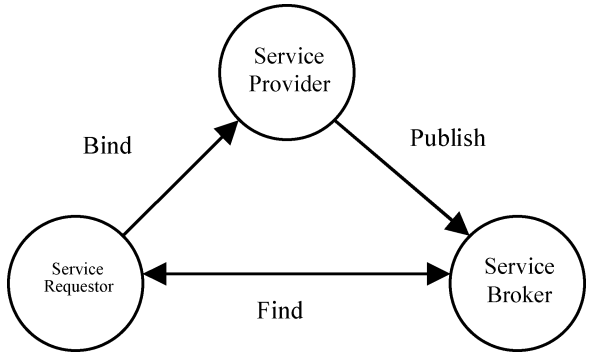

Fig. 1. Service publish-find-bind.

Component systems adhere to the principle of divide and conquer for managing complexity. Components are the smallest possible reusable software. They are self-managing, independent and ideally can be reused in multiple environments. Through a components' interface, it is possible to compose a component with other components and integrate this composite component into a system in a straightforward way. Composite component architectures are formed from layered components where components at the lower layers provide services to the components above them, through middleware.

An interface represents the point of interaction or communication between two software entities. In both component and service development, the notion and concept of an interface is the key to a successful implementation. However, SOAs abstract the component interfaces further, in order to expose a particular service. The following are the key interface-related definitions.

— **Interface**: An interface defines a set of public method signatures that are logically grouped. An interface defines an entry point, as well as the contract between the provider/consumer of a service.
— **Published interface**: An interface that is uniquely identifiable and made available through a registry for clients to dynamically discover.
— **Public interface**: An interface that is available for clients to use but is not published, thus requiring static knowledge on the part of the client.
— **Dual interface**: Frequently interfaces are developed as pairs, such that one interface depends on another.

In many ways, the terminology for services is much the same as the terminology used to describe component-based development; however, there are specific terms used to define service elements. As compared to components, the service-oriented approach implies a further abstraction through an additional application architecture layer. A service exposes an external view of a system, while having internal reuse and composition through the use of components.

There are several key differences between services, components and objects, and many shared concepts.

Services
— Language Independent
— Platform Independent

— Self Describing
— Location Transparent

Components

— Contain Infrastructure Services
— Language Independent
— Platform Dependent

Objects

— Contain Underlying Logic
— Language Dependent
— Platform Dependent

Recent Gartner research has highlighted the importance of service-oriented architectures in solving the business integration issue. There is a growing realization that integration is best solved using a service-oriented architecture.

B. Service Integration and Interaction

A service topology is a pattern for composing a distributed application where the

— **physical network** identifies the physical infrastructure connecting the computing devices;
— **logical network** utilizes a point-to-point topology that abstracts the participants from the physical infrastructure.

The service topology abstracts the participants from both the physical and logical network. Two key service interaction topologies exist.

— **Connectors**: Connectors direct the output of one service into the input of another service in a point-to-point fashion where the participants are directly interconnected with each other rather than using a centralized intermediary.
— **Orchestrations**: Orchestrations utilize a high-level scripting language to control the sequence and flow of service execution, using a hub and spoke topology. Hub & Spoke is a topology that refers to the use of a central hub to coordinate interacting services. The orchestration server may use an orchestration script to govern the interactions that will occur between services.

Research has shown that in order to overcome information sharing issues we must look into the use of structured document formats. The Extensible Markup Language (XML) and Resource Description Framework (RDF) have both proved to be suitable in overcoming such issues. Both XML and RDF provide an information sharing medium for system interoperation through structured data formats The formalization of the representation of these data structures is made possible using an XML-Schema. XML-Schema is a W3C recommendation that places rules that XML documents must adhere to in order to be validated and accepted for further processing.

Just-in-time integration can be used to connect or orchestrate services. Just-in-time integration among collaborations of web services enables available services to be bound dynamically during runtime. In this case a service consumer describes the required capabilities of a service and uses a services directory to locate such a service. Once a service has been located, just-in-time integration can be used to bind to it.

Businesses are not required to replace their existing infrastructures in order to adopt a service framework. Service implementations benefit from the ability to wrap legacy applications. Wrapping is a technique in which legacy code is given an interface that facilitates interoperation with other software applications that are based on a different technology. Wrapping enables legacy systems to interoperate with newer technologies and services.

Once connection and documents interchange issues are solved, it should be possible to plug-in services that appear on a network into a single information bus. Applications are then composed of a collection of collaborating services. A system is then able to evolve through the addition of new services where each service provides access to coarse-grained business processes.

C. Extended Enterprise

A service network is an application level network that leverages the concepts of SOA. The service network is composed of service consumers and providers. A service network is made possible by the standardization of service interaction. This model of integrating corporations via service is often called the extended enterprise. The extended enterprise is a concept that promotes the use of application level interaction between companies to enable near real-time integration. Thus, applications at one company become extensions to the applications at their trading partners, and vice-versa. The extended enterprise leverages the concepts of SOA as infrastructure for collaborating parties.

D. Web Services

Service-oriented methodologies are currently employed in many system design approaches, in particular among business collaboration systems. An emerging technology that aims at addressing some of the service integration complexity issues is web services. Several major players in the e-business development community including IBM, Microsoft, Hewlett Packard, Compaq, and Userland, have adopted this approach.

The web service technology is based on the SOA concepts, focusing on the interactions of services over networks. It is widely considered to be the next-generation distributed computing model. Web services are an evolution in the way we architect, design, implement and deploy service-oriented systems. A web service is a software application that is network distributed and identified by a URI. Web service interfaces and binding are capable of being defined, described and discovered by XML artefacts.

The guiding principles for Web Services are

— service-oriented architecture;
— standardized specification and interaction;
— use of meta-languages.

Web services enable collaborating business-trading partners to interact in a straightforward way via the Extended Enterprise.

Web services leverage three key technologies.

— **Universal Description Discovery and Integration (UDDI)**: Service repository.
— **Web Service Definition Language (WSDL)**: Service Interface Definition.

— **Simple Object Access Protocol (SOAP)**: Interaction Protocol.

1) Service Repository: UDDI is a global meta directory for locating web services by enabling robust queries against rich metadata. UDDI is an industry specification for building flexible, interoperable XML Web service registries. It provides a means to describe trading partners and the services they provide. UDDI is often viewed as the "yellow-pages" of web services. Registries (also called directories), are repositories where information about trading partner profiles or services is stored. A trading partner profile is a digital method for determining how, and to what extent, a company can trade online. The profile may include technical capabilities, security requirements, transports, and message correlation mechanisms, as well as the business dialects spoken. The profile serves as a basis for creating a trading agreement. The agreement is a digital agreement that specifies how the participants will conduct online business.

2) Service Interface Definition: WSDL is an XML format for describing network service interfaces that operate through the use of 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. WSDL is extensible to allow description of services and their messages regardless of what message formats or network protocols are used to communicate.

WSDL descriptions contain the following.

— **Types**: a container for data type definitions using some type system.
— **Message**: an abstract, typed definition of the data being communicated.
— **Operation**: an abstract description of an action supported by the service interface.
— **Port Type**: an abstract set of operations supported by the service.
— **Binding**: a concrete protocol and data format specification for a particular port.
— **Port**: a service interface defined as a combination of a binding and a network address.
— **Service**: a collection of related service interfaces.

In many cases, the WSDL for a service is stored in the service repository. Registries can be public (open to everyone), or private (available to a single company or to a trading community).

3) Interaction Protocol: SOAP is a lightweight protocol intended for exchanging structured information in a decentralized, distributed environment. SOAP is responsible for the transport and delivery of web service invocations. SOAP messages are sent between service consumer and service provider to bind these services. SOAP messages are usually transported over the widely used HTTP protocol. A SOAP Server is a server that hosts web services and acts as a provider of services. SOAP Servers have support for SOAP as well as WSDL. The SOAP Server is the primary participant in a Service Network

4) Types of Web Services: Conceptually, we can create and utilize two types of web services.

Programmatic web service—Programmatic web services provide their services in the form of operations without a user interface. They are integrated into applications that have existing user interface tiers. These types of web services enable the developer at the requesting end to customize a user interface specifically for their business requirements. A conceptual model for programmatic web services is shown in Fig. 2.

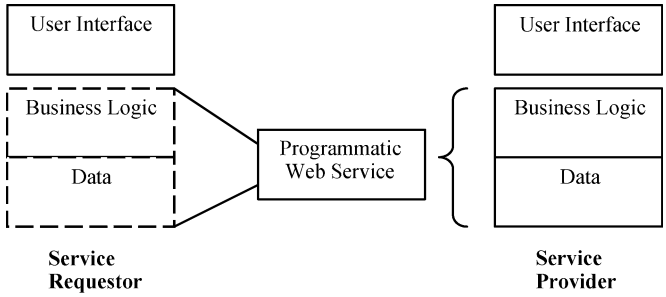

Fig. 2. Programmatic web service.

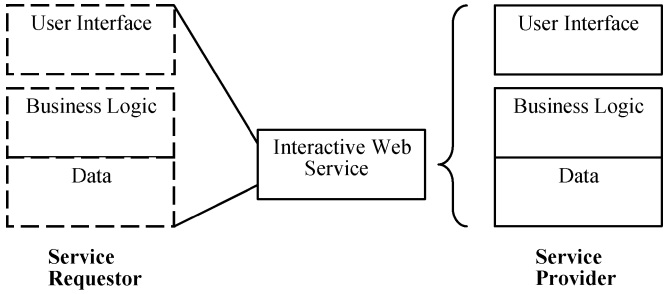

Fig. 3. Interactive web service.

Interactive web services—An interactive web service exposes its provided services in the form of operations along with a user interface. These types of web services simplify the development of applications at the service requestor end, but may limit business workflow integration. A conceptual model for interactive web services is shown in Fig. 3.

The advent of B2B collaborations has created avenues for the SMEs to form consortia as a competitive approach.

Previous practices in system development have tended to produce monolithic, tightly coupled applications, and subsystems that are susceptible to bugs as modifications are made to any one subsystem.

Recent developments, such as web services facilitate decentralized, loosely coupled systems that are capable of dynamic binding. These types of systems provide flexibility with their just-in-time integration of services and have been widely accepted by the e-business community. However, much of the software industry's focus so far has been on the underlying technology for implementing Web services and their interactions. There has been very little attention to the appropriate practices and tools required for constructing enterprise level software solutions that are composed of collaborating services. While web services provide a framework that facilitates point-to-point integration of services, on their own they do not provide a framework for service plug and play. This technology does not fully support end-to-end development of enterprise

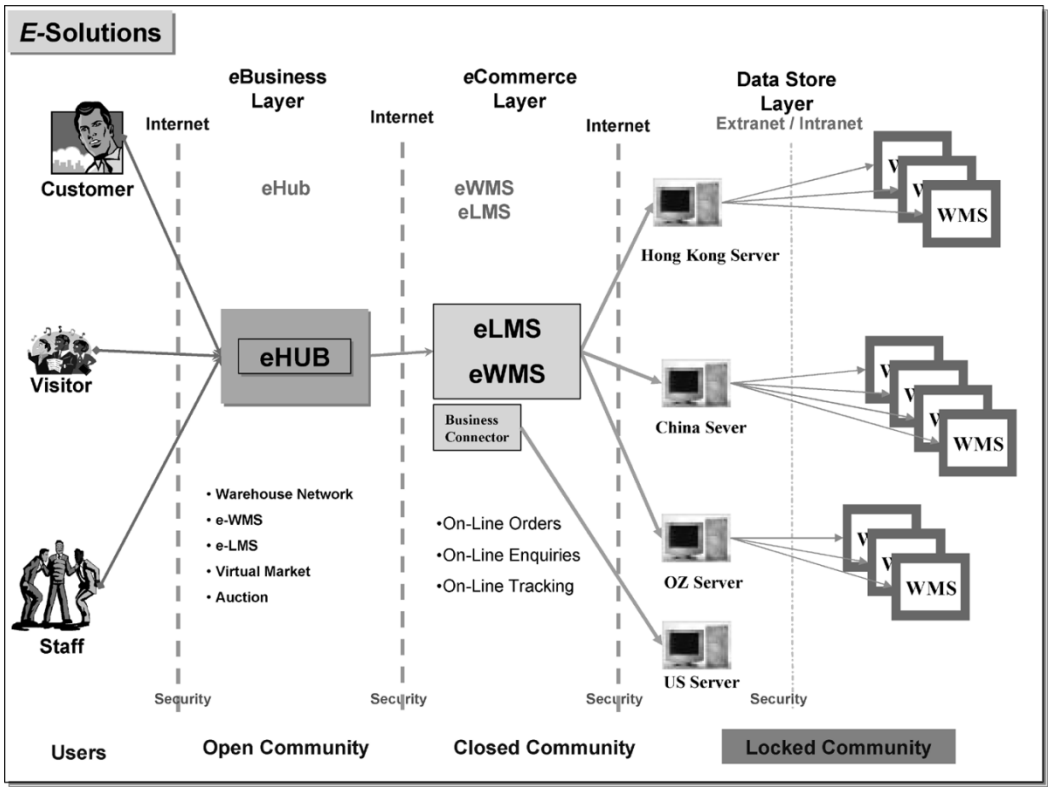

Fig. 4. e-Hub architecture (one eHub is shown).

applications that are composed out of collaborating services. The complexity of service integration and reconfiguration has burdened the adoption of such systems.

In order to achieve a warehousing and transportation network that is geographically widely distributed and provides a greater range of services, consortia partners require a new flexible information technology framework that facilitates straight forward service integration and reconfiguration in a cost effective manner. We propose a distributed Meta model driven framework that eases service integration and reconfiguration, and redirects this task away from the software developer to the softwares user.

IV. Architectural Requirements

Logistics can basically be separated into the following.

- **Transportation logistics**, which is the movement of goods to the right place in the right quantities.
- **Warehousing and storage logistics**, which is the storage of goods at the right place in order to meet uncertain conditions and demand.

The overall distributed supply chain application architecture consists of one or more eHubs that are connected to different companies in their own geographical area of operation. The schematic for the architecture shown in Fig. 4 illustrates the interactions of the different communities (one eHub is shown). It is intended that the architecture would be layered so that each corporation would at any time be able to replace its electronic Warehouse Management System (eWMS) or electronic Logistics Management System (eLMS) layers without affecting the rest of the system.

V. Distributed Meta Framework

As shown in Fig. 5, the presented framework explicitly identifies an enterprise service. Through its interfaces an enterprise service exposes both its provided and required operations. In this way it is possible to make changes to any of the connected service internals as long as the changes continue to adhere to the interface contract. An interface contract specifies the pre-conditions that must be met by a client prior to invoking a server operation and the post-conditions that it will receive as a return from the server. We have used the Unified Modeling Language (UML) to illustrate our framework. However, in the future it may be necessary to define a different notation system.

A. Interfaces

An interface is a collection of operations that specify a service. Interfaces are seen as a provider and a consumer [12]. The concept of required and provided interfaces is essential to enabling software plug and play. It is possible to replace or modify a services' internals as long as the implementation adheres to the interface contract. An interface contract specifies the service interface, its exposed attributes and operations, the interaction protocol and the pre and post-conditions that must be met in order for enterprise applications to interoperate. Each enterprise application will expose multiple services and each service

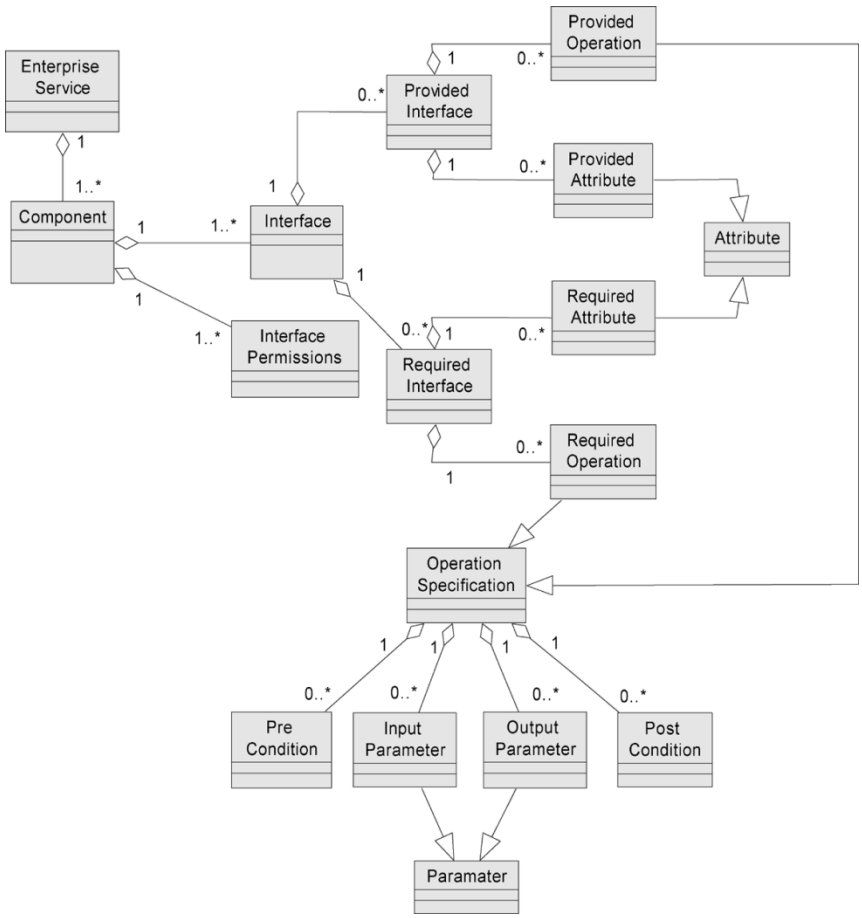

Fig. 5. Enterprise service.

will expose one or more provided interfaces and zero or more required interfaces.

A services' provided and required interfaces can be obtained by querying a central Meta data repository or the web-service itself.

- **Provided interfaces** expose the underlying provided operations and attributes that represent the services that a service provides. Services are operations that can be accessed through the provided interface by a client. The results of a call to a provided operation should be documented as post-conditions of that operation.
- **Required interfaces**. A service may also request a list of services that it requires in order to perform. A service may request either operations or attributes through its required interfaces. The requirements of a web-service whether attributes or operations should be documented in the components pre-conditions.

Operations and attributes are either provided or required by a component.

- **Operation**. An operation is the implementation of a specific service function that represents the dynamic behavior of that service.
- **Attributes**. An attribute is a named property value that describes the characteristics of a service.

B. Service Security

Access to enterprise services is restricted through a certificate scheme. As a part of a clients' request, the client must provide a certificate. Based on this certificate certain, interfaces are hidden and disabled. In conventional qualification based classification, the means is a secret ticket that permits permission to anything presenting it, and so the qualifications are required to be rigorously restricted. A SPKI [13] certificate format focuses on authorization rather than authentication. Specific authorization is granted to a public key, without necessarily requiring identity of the holder of the corresponding private key. The public key is used to identify the key holder. Consequently, it does not need to be treated as secret and to be strictly controlled.

The following items are stored in the SPKI certificate (Fig. 6) [14].

- **Version**: The version number.
- **Serial number**: A unique number for identifying the certificate.
- **Signature algorithms identifier**: Information on the signature algorithm.
- **Issuer**: The certification authority that issues the certificate.
- **Validity period**: The period of time during which the certificate is valid.
- **Subject**: The owner of the public key.

— **Subject public key information**: Information on the subject's public key.
— **Issuer unique identifier**: Optional field for identifying the issuer uniquely.
— **Subject unique identifier**: Optional field for identifying the subject uniquely.
— **Extension fields**: This is for including additional data.
— **Digital signature**: The digital signature for the above fields.

In this way, it is possible to do the following.

— Expose only a proportion of an enterprise service's operations by customizing the privileges of its provided interface through the use of certificates. This makes it possible to provide varying grades of functionality based on, for example, the price paid by the client user.
— Restrict the users of an enterprise service through a certificate scheme; thus, security is improved for sensitive data.
— Introduce a special developer portal that can offer developers access to the code, compilers, GUI tools, and documentation they need to do their jobs. Testing and debugging can be performed remotely on special testing interfaces not permitted for use by normal users of the enterprise service. In this way, it is possible to monitor and classify bugs and performance issues associated with each service.

C. Configuration—Attribute Connectors

Attribute connectors (Fig. 7) are used to glue required and provided attributes together. Once glued, a required attribute always requests the value it requires from the provided attribute that it is glued to. Attribute connectors can be used to connect required service attributes to provided ones, or to a static value provided by the user. An attribute adaptor can be used to convert a provided attribute that is of the wrong type so that it satisfies the requirements of a required attribute [15]. Attribute adaptors are not discussed in this paper.

D. Composition—Operation Connectors

Like attribute connectors, operation connectors (Fig. 8) are used to glue required and provided operations together. Once glued, a required operation always calls the provided operations that it is glued to. Operation connectors can be used to connect required service operations to provided ones or to a static return value as provided by the user. Operation adaptors may be used when a provided operation has an incompatible return type or parameters and needs conversion [15]. Operation connectors are not discussed in this paper.

VI. Prototype System

We have developed a prototype electronic Warehouse Management System (eWMS) in order to test the feasibility of our approach. The eWMS is an online order capturing and tracking system that allows users to manage their supply-chain. The system is able to dynamically integrate a number of heterogeneous warehouse services. Users of the system are able to perform enquiries about their Lot Balance, Lot Movement, Accounts Payable, and Order Status, as well as being able to submit online orders such as warehouse booking, goods transfer, and goods delivery. The eWMS is aimed at

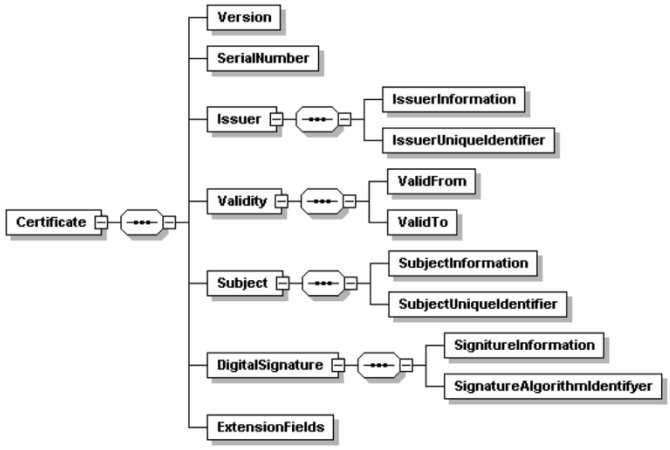

Fig. 6. Certificate XML schema.

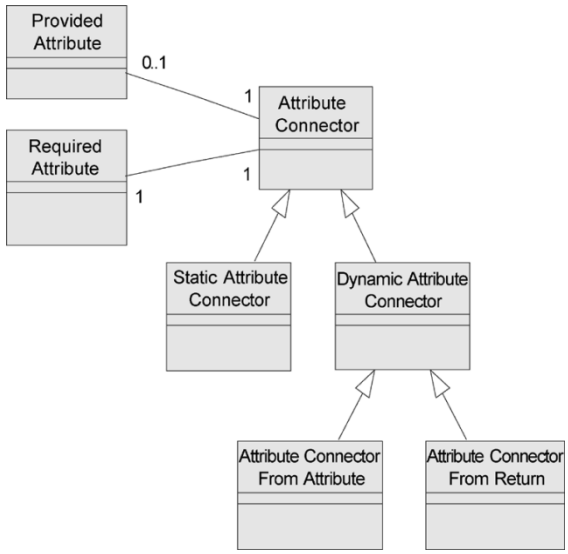

Fig. 7. UML Meta model for component attribute connector.

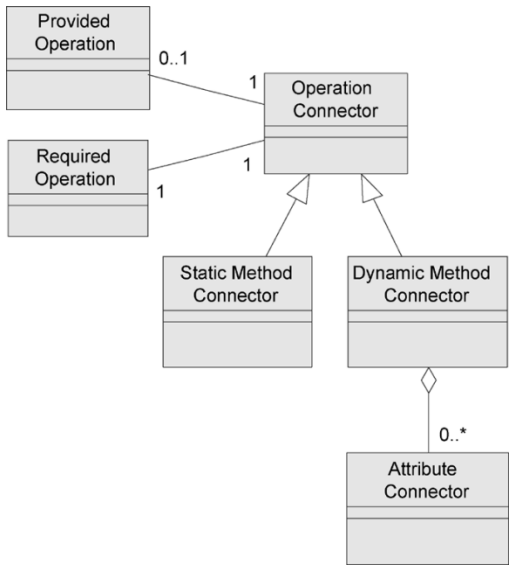

Fig. 8. UML Meta model for component operation connector.

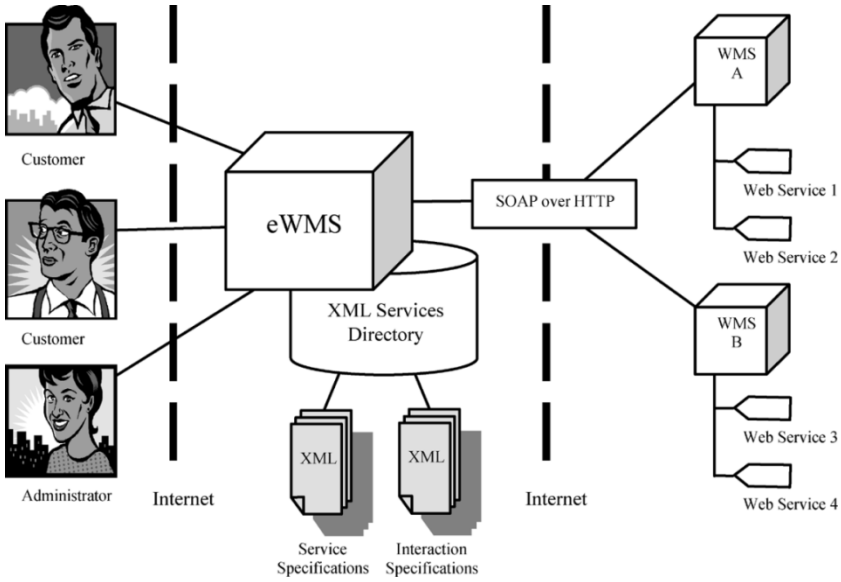

Fig. 9. eWMS architecture overview.

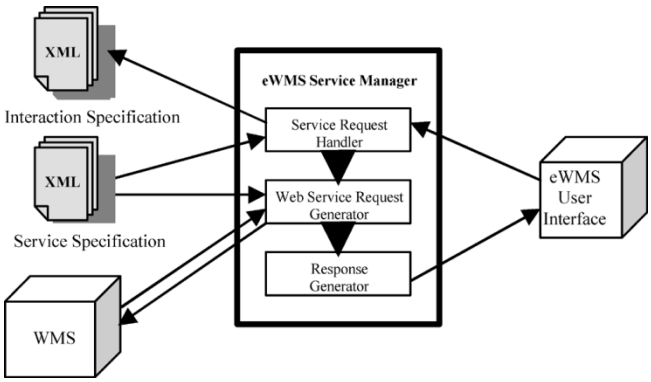

Fig. 10. eWMS service provider architecture.

the extended enterprise by providing an infrastructure for the integration and reconfiguration of services that are provided by warehousing SMEs. An overview of the eWMS architecture is shown in Fig. 9.

A. *eWMS Service Manager*

The eWMS Service Manager deals with service requests from users. It is responsible for the interaction between the available services originating from WMS consortia partners. The eWMS Service Manager refers to the XML service and interaction specifications. This data is used to interconnect consortia partners. Fig. 10 displays the components that make up the eWMS Service Manager.

The eWMS Service Manager utilizes the Service Request Handler, Web Service Request Generator, and the Service Response Generator modules in order to integrate collaborating consortia partners.

— **Service request handler**: The Service request handler is responsible for the handling of eWMS service requests made by the user of the system. It translates these requests and locates the corresponding service and interaction specifications from which a web service request is generated.

— **Web service generator**: The creation of web service requests from the input specifications is carried out by the web service request generator. The web service generator directly communicates with provided WMS services. Once a request has been received from the WMS service it is sent to the response generator.

— **Response generator**: The response generator accepts responses from WMS services and translates them for display on the eWMS user interface.

A Service Integration Manager (Fig. 11) is used as an administration tool for specifying service interaction. The required eWMS service specifications are shown on the left while the provided WMS service specifications are shown on the right. The middle section is used to specify the service interconnections between provider and consumer.

The service and interaction specifications represent two types of information required to implement the dynamic integration of web services:

— required and provided service specification;
— the interaction specification.

Fig. 12 illustrates the relationship between service specifications and interaction specifications.

— **Service specifications**: The service specification is an XML document that defines a web service and its interface.
— **Provided service interface**: A provided service interface is a defined set of service operations provided by a web service.
— **Required service interface**: A required service interface is defined as a set of service operations required by a web service.
— **Service operations**: Service operations are the fine-grained services that a web service provides or requires. Every service operation defined contains information about the input and output parameters as well as the pre- and post-conditions that are associated with the service operation.

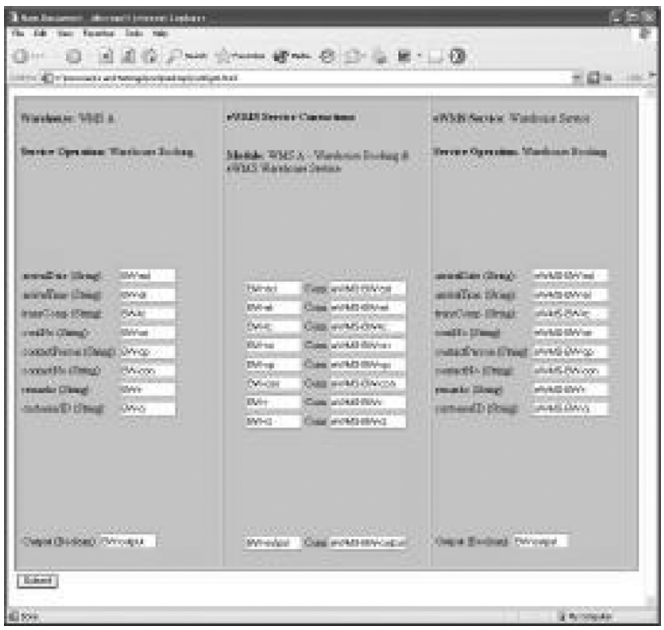

Fig. 11. eWMS service configuration user interface.

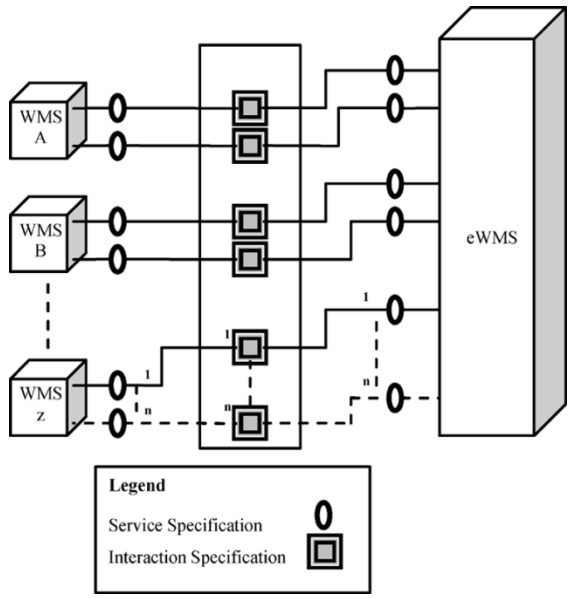

Fig. 12. eWMS enterprise integration conceptual model.

— **Interaction specifications**: Interaction specification documents specify the interactions between required and provided services. A connection between the two services makes up a module specification.

The initial interaction involves the customer selecting a warehouse, in which to do business with. Once the warehouse has been selected, the available web services and their service operations are presented. After selection of the service operation, the customer is presented with an input form generated by the eWMS. Here, the customer can input the required information and submit it to the system for processing. Upon receiving the data submitted from the customer, the eWMS service provider locates the appropriate interaction specifications and interconnection is performed. Not all information required by a connecting service is provided by the customer input. Information may also be provided via other internal eWMS services.

VII. Conclusion

Due to the complexity of integrating heterogeneous enterprise services, the lack of a framework for expressing service collaboration makes such service-oriented software more difficult to integrate, reconfigure, and evolve. In order to take the notion of service integration further, we need to develop a model of inter-enterprise workflow that is capable of consuming services. This paper presents a framework for service integration in an extended logistics enterprise.

References

[1] W. Brenner, R. Zarnekow, and H. Wittig, Eds., *Intelligent Software Agents: Foundations and Applications*. Berlin/Heidelberg, Germany: Springer-Verlag, 1998.
[2] Aglets. website. [Online] Available: www.trl.ibm.co.jp/aglets/
[3] Voyager. website. [Online] Available: www.objectspace.com/voyager/
[4] FTP. website. [Online] Available: www.ftp.com/
[5] Odyssey. website. [Online] Available: www.genmagic.com
[6] JATLite. website. [Online] Available: java.stanford.edu/java_agent/html
[7] T. Sandholm and Q. Huai, "Nomad: mobile agent system for an Internet-based auction house," *IEEE Internet Comput.*, vol. 4, no. 2, pp. 80–86, Mar.-Apr. 2000.
[8] J. M. Andreoli, F. Pacull, and R. Pareshi, "Xpect: a framework for electronic commerce," *IEEE Internet Comput.*, vol. 1, no. 4, pp. 40–48, Jul.-Aug. 1997.
[9] C. Yeung, T. Pang-Fei, and J. Yen, "A multi-agent based Tourism Kiosk on internet," in *Proc. 31st Hawaii Int. Conf. System Sciences 1998*, vol. 4, Kohala Coast, HI, 1998.
[10] P. Dasqupta, N. Narasimhan, L. E. Moser, and P. M. Smith, "MAGNET: Mobile agent for networked electronic trading," *IEEE Trans. Knowl. Data Eng.*, vol. 11, no. 4, pp. 509–525, Jul.-Aug. 1999.
[11] M. Stal, "Web services: Beyond component-based computing," *Commun. ACM*, vol. 45, no. 10, pp. 71–76, Oct. 2002.
[12] N. Medvidovic and R. Taylor, "Separating fact from fiction in software architecture," in *Proc. Int. Software Architecture Workshop*, Orlando, FL, Nov. 1998.
[13] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Ylonen. (1999) SPKI (Simple Public Key Infrastructure). [Online] Available: www.ietf.org/rfc/rfc2693.txt
[14] H. Chan, R. Lee, T. Dillon, and E. Chang, *E-Commerce Fundamentals and Applications'*. New York: Wiley, 2001.
[15] D. Rine, N. Nada, and K. Jaber, "Using adapters to reduce interaction complexity in reusable component-based software development," in *Proc. Symp. Software Reusability*, Los Angeles, CA, May 1999.

Alex Talevski (M'05) has recently completed his Ph.D. thesis, entitled "Reconfigurable Plug and Play Component-Based Software."

His research interests include component-based software engineering, software tailoring and evolution, e-commerce, extended enterprises, and self-adaptive neural networks. He has successfully lead several commercial software development projects through to completion. He has published more than 13 refereed papers in international journals and conferences, two of which have won best paper awards.

Elizabeth Chang (M'93) is a Professor in Information Technology (IT) and Software Engineering at Curtin University of Technology, Perth, WA, Australia. She is the Director of the Centre for Frontier Technology for Extended Enterprises, Curtin Business School, and was Founder of the Centre for IT Applications and Logistics Informatics, Newcastle University, Newcastle upon Tyne, U.K. Her research interests include ontology-based software engineering, object/component-based development trust, security and risks in e-business, XML, web services, P2P and middleware for e-commerce, usability (including web usability) and object-oriented user interface engineering. As a Project Manager, she has successfully managed several commercial-grade IT projects for industry, taking them through the entire software life cycle to project completion. She has published over 100 scientific conference, journal, and numerous invited keynote papers at international conferences and has written two books.

Tharam S. Dillon (M'83–SM'87–F'98) is the Dean of Faculty of Information Technology, University of Technology Sydney (UTS), Sydney, NSW, Australia. He is currently also the Director of the eXel Research Lab. Previously, he was the Foundation Professor of Computer Science and the Director of the Applied Computing Research Institute at La Trobe University, Bundoora, Vic., Australia. He has worked with industry and commerce in developing systems in telecommunications, health care systems, e-commerce, logistics, power systems, and banking and finance. He is Chairman of the IFIP Working Group 2.12 on Web Semantics. He has published more than 400 papers in international and national journals and conferences, has written five books, edited five books, and has published 17 chapters in edited books. His research interests include data mining, internet computing, e-commerce, hybrid neuro-symbolic systems, neural nets, software engineering, database systems, computer networks, and trusted computing.

Dr. Dillon is Co-Editor-in-Chief of three international journals and was an Advisory Editor of the IEEE TRANSACTIONS ON NEURAL NETWORKS for five years. He is on the Advisory Editorial Board of *Applied Intelligence* and *Computer Communications*. He is a Fellow of the Institution of Engineers (Australia) and the Australian Computer Society.